Introduction

This reference manual targets application developers. It provides complete information on how to use the STM32H7R3, STM32H7S3, STM32H7R7, and STM32H7S7 memory and peripherals.

Microcontrollers of the STM32H7R3/7S3 and STM32H7R7/7S7 lines have different memory sizes, packages, and peripherals. They are referred to as STM32H7Rx/7Sx hereafter.

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

For information on the Arm® Cortex®-M7 with FPU core, refer to the corresponding Arm technical reference manual available from www.arm.com.

Related documents

- Cortex®-M7 programming manual (PM0253)
- STM32H7S3x8 and STM32H7S7x8 (DS14359)
- STM32H7R3x8 and STM32H7R7x8 (DS14360)
## Contents

1 **Documentation conventions** ................................................. 120  
   1.1 General information .................................................. 120  
   1.2 List of abbreviations for registers ................................. 120  
   1.3 Glossary ............................................................... 121  
   1.4 Availability of peripherals ........................................... 121  
   1.5 Availability of security features ................................. 121  

2 **Memory and bus architecture** ......................................... 122  
   2.1 System architecture .................................................. 122  
      2.1.1 Bus matrices .................................................... 124  
      2.1.2 Bus-to-bus bridges ............................................. 124  
      2.1.3 TCM-buses ....................................................... 124  
      2.1.4 CPU buses ....................................................... 124  
      2.1.5 Bus master peripherals ....................................... 125  
      2.1.6 GPU2D-bus ....................................................... 126  
      2.1.7 GFXMMU-bus ..................................................... 126  
      2.1.8 Clocks to functional blocks ................................... 126  
   2.2 AXI interconnect matrix (AXIM) ................................... 127  
      2.2.1 AXI introduction ................................................ 127  
      2.2.2 AXI interconnect main features ............................... 127  
      2.2.3 AXI interconnect functional description .................... 128  
      2.2.4 AXI interconnect registers ................................... 131  
      2.2.5 AXI interconnect register map ................................ 139  
   2.3 Memory organization .................................................. 149  
      2.3.1 Introduction ..................................................... 149  
      2.3.2 Memory map and register boundary addresses ............... 150  
   2.4 Embedded SRAM ....................................................... 159  
   2.5 Flash memory overview .............................................. 162  
   2.6 Boot configuration ................................................... 162  

3 **RAMECC monitoring (RAMECC)** ....................................... 164  
   3.1 Introduction ......................................................... 164  
   3.2 RAMECC main features .............................................. 164
3.3 RAMECC functional description .................................................. 164
3.3.1 RAMECC internal signals ....................................................... 166
3.3.2 RAMECC monitor mapping ..................................................... 167
3.3.3 ECC fault injection test on SRAM1, SRAM3, and TCM ................. 168
3.3.4 Read access latency .............................................................. 168
3.4 RAMECC low-power modes ....................................................... 168
3.5 RAMECC interrupts ................................................................. 169
3.6 RAMECC registers ................................................................. 170
3.6.1 RAMECC interrupt enable register (RAM ECC_IER) ..................... 170
3.6.2 RAMECC monitor x configuration register
      (RAMECC_MxCR) .............................................................. 171
3.6.3 RAMECC monitor x status register
      (RAMECC_MxSR) .............................................................. 173
3.6.4 RAMECC monitor x failing address register
      (RAMECC_MxFAR) ............................................................ 173
3.6.5 RAMECC monitor x failing data low register
      (RAMECC_MxFDRL) .......................................................... 174
3.6.6 RAMECC monitor x failing data high register
      (RAMECC_MxFDRH) .......................................................... 174
3.6.7 RAMECC monitor x failing ECC error code register
      (RAMECC_MxFECR) .......................................................... 175
3.6.8 RAMECC register map ......................................................... 176

4 System security ................................................................. 177
4.1 Key security features ............................................................. 177
4.2 Secure install ..................................................................... 178
4.3 Secure boot ........................................................................ 179
4.3.1 Unique boot entry ............................................................ 179
4.3.2 Security services in system flash memory ................................ 180
4.4 Secure update ................................................................. 182
4.5 Resource isolation using hide protect levels ............................. 182
4.5.1 Temporal isolation using secure hide protection (HDP) ............. 182
4.6 Secure execution ............................................................... 183
4.6.1 Memory protection unit (MPU) ............................................ 183
4.6.2 Embedded flash memory write protection .............................. 183
4.6.3 Tamper detection and response ......................................... 183
4.7 Secure storage ................................................................. 186
4.7.1 Hardware secret key management ....................................... 187
5.4.7 Description of security option bytes ........................................... 228

5.5 FLASH protection mechanisms .................................................... 230
5.5.1 FLASH configuration protection .............................................. 230
5.5.2 Write protection ................................................................. 231
5.5.3 Life cycle management .......................................................... 232
5.5.4 Hide protected system flash area .............................................. 233
5.5.5 Hide protected user flash area ................................................ 233

5.6 FLASH low-power modes ............................................................ 236
5.6.1 Introduction ............................................................................ 236
5.6.2 Managing the FLASH domain switching to Stop or Standby .......... 236

5.7 FLASH error management ............................................................ 237
5.7.1 Introduction ............................................................................ 237
5.7.2 Write protection error (WRPERRF) ........................................... 237
5.7.3 Programming sequence error (PGSERRF) ................................. 238
5.7.4 Strobe error (STRBERRF) ......................................................... 238
5.7.5 Inconsistency error (INCERRF) .................................................. 239
5.7.6 Error correction code error (SNECCERRF/DBECCERRF) ............ 239
5.7.7 Read secure error (RDSERRF) ................................................... 240
5.7.8 CRC read error (CRCRDERRF) ................................................ 240
5.7.9 Option byte change error (OPTERRF) ....................................... 240
5.7.10 Key valid error (KVEF) ............................................................ 241
5.7.11 Key transfer error (KTEF) ......................................................... 241
5.7.12 Option byte loading error (OBLERRF) .................................... 241
5.7.13 Miscellaneous HardFault errors .............................................. 242

5.8 FLASH interrupts ......................................................................... 242

5.9 FLASH registers ......................................................................... 244
5.9.1 FLASH access control register (FLASH_ACR) .......................... 244
5.9.2 FLASH control key register (FLASH_KEYR) ............................... 245
5.9.3 FLASH control register (FLASH_CR) ......................................... 245
5.9.4 FLASH status register (FLASH_SR) .......................................... 248
5.9.5 FLASH interrupt enable register (FLASH_IER) ........................... 250
5.9.6 FLASH interrupt status register (FLASH_ISR) ........................... 251
5.9.7 FLASH interrupt clear register (FLASH_ICR) .............................. 253
5.9.8 FLASH CRC control register (FLASH_CRCRR) ......................... 254
5.9.9 FLASH CRC start address register (FLASH_CRCADDR) .......... 255
5.9.10 FLASH CRC end address register (FLASH_CRCADDR) .......... 256
## Power control (PWR)

6.1 Introduction ................................................. 278
6.2 PWR main features ........................................... 278
6.3 PWR block diagram ........................................... 279
6.3.1 PWR pins and internal signals .......................... 280
6.4 Power supplies .............................................. 282
6.4.1 System supply startup ..................................... 286
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.2</td>
<td>Core domain</td>
<td>289</td>
</tr>
<tr>
<td>6.4.3</td>
<td>PWR external supply</td>
<td>291</td>
</tr>
<tr>
<td>6.4.4</td>
<td>Backup domain</td>
<td>291</td>
</tr>
<tr>
<td>6.4.5</td>
<td>VBAT battery charging</td>
<td>293</td>
</tr>
<tr>
<td>6.4.6</td>
<td>Analog supply</td>
<td>294</td>
</tr>
<tr>
<td>6.4.7</td>
<td>USB regulator</td>
<td>295</td>
</tr>
<tr>
<td>6.5</td>
<td>Power supply supervision</td>
<td>296</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Power-on reset (POR)/power-down reset (PDR)</td>
<td>296</td>
</tr>
<tr>
<td>6.5.2</td>
<td>Brownout reset (BOR)</td>
<td>297</td>
</tr>
<tr>
<td>6.5.3</td>
<td>Programmable voltage detector (PVD)</td>
<td>298</td>
</tr>
<tr>
<td>6.5.4</td>
<td>Analog voltage detector (AVD)</td>
<td>299</td>
</tr>
<tr>
<td>6.5.5</td>
<td>Battery voltage thresholds</td>
<td>300</td>
</tr>
<tr>
<td>6.5.6</td>
<td>Temperature thresholds</td>
<td>301</td>
</tr>
<tr>
<td>6.5.7</td>
<td>VCORE maximum voltage level detector</td>
<td>302</td>
</tr>
<tr>
<td>6.6</td>
<td>Power management</td>
<td>303</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Operating modes</td>
<td>303</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Voltage scaling</td>
<td>304</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Power management examples</td>
<td>305</td>
</tr>
<tr>
<td>6.7</td>
<td>Low-power modes</td>
<td>308</td>
</tr>
<tr>
<td>6.7.1</td>
<td>Slowing down system clocks</td>
<td>309</td>
</tr>
<tr>
<td>6.7.2</td>
<td>Controlling peripheral clocks</td>
<td>309</td>
</tr>
<tr>
<td>6.7.3</td>
<td>Entering low-power modes</td>
<td>309</td>
</tr>
<tr>
<td>6.7.4</td>
<td>Exiting from low-power modes</td>
<td>309</td>
</tr>
<tr>
<td>6.7.5</td>
<td>Sleep mode</td>
<td>311</td>
</tr>
<tr>
<td>6.7.6</td>
<td>Stop mode</td>
<td>311</td>
</tr>
<tr>
<td>6.7.7</td>
<td>Standby mode</td>
<td>314</td>
</tr>
<tr>
<td>6.7.8</td>
<td>Monitoring low-power modes</td>
<td>316</td>
</tr>
<tr>
<td>6.8</td>
<td>PWR registers</td>
<td>317</td>
</tr>
<tr>
<td>6.8.1</td>
<td>PWR control register 1 (PWR_CR1)</td>
<td>317</td>
</tr>
<tr>
<td>6.8.2</td>
<td>PWR control status register 1 (PWR_SR1)</td>
<td>319</td>
</tr>
<tr>
<td>6.8.3</td>
<td>PWR control status register 1 (PWR_CSR1)</td>
<td>320</td>
</tr>
<tr>
<td>6.8.4</td>
<td>PWR control register 2 (PWR_CSR2)</td>
<td>322</td>
</tr>
<tr>
<td>6.8.5</td>
<td>PWR CPU control register 3 (PWR_CSR3)</td>
<td>325</td>
</tr>
<tr>
<td>6.8.6</td>
<td>PWR control status register 4 (PWR_CSR4)</td>
<td>326</td>
</tr>
<tr>
<td>6.8.7</td>
<td>PWR wakeup clear register (PWR_WKUPCR)</td>
<td>327</td>
</tr>
<tr>
<td>6.8.8</td>
<td>PWR wakeup flag register (PWR_WKUPFR)</td>
<td>328</td>
</tr>
<tr>
<td>6.8.9</td>
<td>PWR wakeup enable and polarity register (PWR_WKUPEPR)</td>
<td>329</td>
</tr>
</tbody>
</table>
6.8.10 PWR USB Type-C and Power Delivery register (PWR_UCPDR) ... 330
6.8.11 PWR apply pull configuration register (PWR_APCR) ............... 331
6.8.12 PWR port N pull-up control register (PWR_PUCRN) ............... 333
6.8.13 PWR port N pull-down control register (PWR_PDCRN) ............ 334
6.8.14 PWR port O pull-up control register (PWR_PUCRO) ............... 335
6.8.15 PWR port O pull-down control register (PWR_PDCRO) .......... 336
6.8.16 PWR port P pull-down control register (PWR_PDCRP) .......... 337
6.8.17 PWR register map ............................................. 338

7 Reset and clock control (RCC) ............................................. 340
7.1 RCC main features .................................................. 340
7.2 RCC block diagram .................................................. 341
7.3 RCC pins and internal signals ....................................... 342
7.4 RCC reset block functional description ............................. 344
  7.4.1 Reset from PWR block ........................................... 344
  7.4.2 The system and application resets (sys_rst and nreset) ....... 346
  7.4.3 The NRST reset .................................................. 346
  7.4.4 Low-power mode security reset (lpwr_rst) ..................... 346
  7.4.5 Backup domain reset ........................................... 347
  7.4.6 Coresight debug reset .......................................... 347
  7.4.7 Option bytes loading ........................................... 348
  7.4.8 Peripheral resets .............................................. 348
  7.4.9 Reset coverage summary ...................................... 348
  7.4.10 Reset source identification .................................. 350
  7.4.11 Power-on and wakeup sequences ................................ 350
7.5 RCC clock block functional description ............................. 352
  7.5.1 Clock naming convention ...................................... 354
  7.5.2 Oscillators description ........................................ 354
  7.5.3 Clock security system (CSS) ................................... 364
  7.5.4 Clock output generation (MCO1/MCO2) ........................ 367
  7.5.5 PLL description ................................................ 369
  7.5.6 System clock (sys_ck) ......................................... 375
  7.5.7 Clock protection ................................................ 378
  7.5.8 Clock generation in Stop and Standby modes .................. 379
  7.5.9 Peripheral clock distribution .................................. 381
  7.5.10 General clock concept overview ............................... 410
  7.5.11 Peripheral allocation ......................................... 411
7.5.12 Peripheral clock gating control ............................................. 412
7.5.13 CPU and bus matrix clock gating control ................................. 415
7.5.14 Low-power emulation modes ............................................... 416

7.6 RCC interrupts ................................................................. 417

7.7 RCC Programming examples .................................................. 418
7.7.1 PLL programming procedure ................................................ 418
7.7.2 Frequency configuration examples ...................................... 420

7.8 RCC registers ................................................................. 423
7.8.1 RCC source control register (RCC_CR) ................................. 423
7.8.2 RCC clock protection register (RCC_CKPROTR) ...................... 427
7.8.3 RCC HSI calibration register (RCC_HSICFGR) ......................... 428
7.8.4 RCC clock recovery RC register (RCC_CRRCR) ....................... 429
7.8.5 RCC CSI calibration register (RCC_CSCFGR) ......................... 429
7.8.6 RCC clock configuration register (RCC_CFFGR) ..................... 430
7.8.7 RCC CPU domain clock configuration register (RCC_CDCFGR) ...... 433
7.8.8 RCC AHB clock configuration register (RCC_BMCFGR) .............. 434
7.8.9 RCC APB clocks configuration register (RCC_APCFGR) .............. 435
7.8.10 RCC PLLs clock source selection register (RCC_PLLCKSELR) ...... 437
7.8.11 RCC PLLs configuration register (RCC_PLLCFGR) .................. 439
7.8.12 RCC PLL1 dividers configuration register 1 (RCC_PLL1DIVR1) .... 442
7.8.13 RCC PLL1 dividers configuration register 2 (RCC_PLL1DIVR2) .... 444
7.8.14 RCC PLL1 fractional divider register (RCC_PLL1FRACR) .......... 444
7.8.15 RCC PLL1 Spread Spectrum Clock Generator register 
(RCC_PLL1SSCGR) ................................................................. 446
7.8.16 RCC PLL2 dividers configuration register 1 (RCC_PLL2DIVR1) .... 447
7.8.17 RCC PLL2 dividers configuration register 2 (RCC_PLL2DIVR2) .... 449
7.8.18 RCC PLL2 fractional divider register (RCC_PLL2FRACR) .......... 450
7.8.19 RCC PLL2 Spread Spectrum Clock Generator register 
(RCC_PLL2SSCGR) ................................................................. 451
7.8.20 RCC PLL3 dividers configuration register 1 (RCC_PLL3DIVR1) .... 452
7.8.21 RCC PLL3 dividers configuration register 2 (RCC_PLL3DIVR2) .... 454
7.8.22 RCC PLL3 fractional divider register (RCC_PLL3FRACR) .......... 454
7.8.23 RCC PLL3 Spread Spectrum Clock Generator register 
(RCC_PLL3SSCGR) ................................................................. 456
7.8.24 RCC AHB peripheral kernel clock selection register (RCC_CCIPR1) . 457
7.8.25 RCC APB1 peripherals kernel clock selection register 
(RCC_CCIPR2) ................................................................. 460
7.8.26 RCC APB2 peripherals kernel clock selection register (RCC_CCR3) ........................................ 462
7.8.27 RCC APB4,5 peripherals kernel clock selection register (RCC_CCR4) ........................................ 464
7.8.28 RCC clock source interrupt enable register (RCC_CIER) ....................................................... 466
7.8.29 RCC clock source interrupt flag register (RCC_CIFR) ............................................................... 467
7.8.30 RCC clock source interrupt clear register (RCC_CICR) ............................................................ 469
7.8.31 RCC Backup domain control register (RCC_BDCR) ................................................................. 471
7.8.32 RCC clock control and status register (RCC_CSR) ................................................................. 473
7.8.33 RCC AHB1 peripheral reset register (RCC_AHB1RSTR) ............................................................ 474
7.8.34 RCC AHB2 peripheral reset register (RCC_AHB2RSTR) ............................................................ 475
7.8.35 RCC AHB3 peripheral reset register (RCC_AHB3RSTR) ............................................................ 476
7.8.36 RCC AHB4 peripheral reset register (RCC_AHB4RSTR) ............................................................ 477
7.8.37 RCC AHB5 peripheral reset register (RCC_AHB5RSTR) ............................................................ 479
7.8.38 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1) ......................................................... 481
7.8.39 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2) ......................................................... 484
7.8.40 RCC APB2 peripheral reset register (RCC_APB2RSTR) ............................................................. 485
7.8.41 RCC APB4 peripheral reset register (RCC_APB4RSTR) ............................................................. 487
7.8.42 RCC APB5 peripheral reset register (RCC_APB5RSTR) ............................................................. 489
7.8.43 RCC AXI clocks gating disable register (RCC_CKGDISR) ....................................................... 490
7.8.44 RCC Reset status register (RCC_RSR) .................................................................................... 494
7.8.45 RCC AHB1 clock enable register (RCC_AHB1ENR) ................................................................. 496
7.8.46 RCC AHB2 clock enable register (RCC_AHB2ENR) ................................................................. 498
7.8.47 RCC AHB3 clock enable register (RCC_AHB3ENR) ................................................................. 499
7.8.48 RCC AHB4 clock enable register (RCC_AHB4ENR) ................................................................. 500
7.8.49 RCC AHB5 clock enable register (RCC_AHB5ENR) ................................................................. 502
7.8.50 RCC APB1 clock enable register 1 (RCC_APB1ENR1) ............................................................. 504
7.8.51 RCC APB1 clock enable register 2 (RCC_APB1ENR2) ............................................................. 507
7.8.52 RCC APB2 clock enable register (RCC_APB2ENR) ................................................................. 508
7.8.53 RCC APB4 clock enable register (RCC_APB4ENR) ................................................................. 509
7.8.54 RCC APB5 clock enable register (RCC_APB5ENR) ................................................................. 512
7.8.55 RCC AHB5 low-power clock enable register (RCC_AHB5LPENR) ........................................... 513
7.8.56 RCC AHB1 low-power clock enable register (RCC_AHB1LPENR) ........................................... 515
7.8.57 RCC AHB2 low-power clock enable register (RCC_AHB2LPENR) ........................................... 517
7.8.58 RCC AHB3 low-power clock enable register (RCC_AHB3LPENR) ........................................... 518
7.8.59 RCC AHB4 low-power clock enable register (RCC_AHB4LPENR) ........................................... 519
7.8.60 RCC APB1 low-power clock enable register 1 (RCC_APB1LPENR1) ...................................... 521
7.8.61 RCC APB1 low-power clock enable register 2 (RCC_APB1LPENR2) ...................................... 524
8 System configuration, boot and security (SBS) .............................. 539
  8.1 SBS introduction .......................................................... 539
  8.2 SBS main features .......................................................... 539
  8.3 SBS functional description .............................................. 540
    8.3.1 SBS block diagram .................................................... 540
    8.3.2 SBS internal signals .................................................. 540
    8.3.3 SBS reset and clocks ................................................ 540
    8.3.4 SBS hide protection management ................................... 541
    8.3.5 SBS boot control ...................................................... 541
    8.3.6 SBS debug management .............................................. 542
    8.3.7 SBS voltage booster for I/O analog switches .................... 543
    8.3.8 SBS I/O compensation cell management ........................... 543
    8.3.9 SBS registers access control ...................................... 544
    8.3.10 SBS error management .............................................. 545
  8.4 SBS interrupts ............................................................. 545
  8.5 SBS registers .............................................................. 545
    8.5.1 SBS boot status register (SBS_BOOTSR) ............................ 545
    8.5.2 SBS hide protection control register (SBS_HDPLCR) ............. 546
    8.5.3 SBS hide protection status register (SBS_HDPXSR) ............... 546
    8.5.4 SBS debug control register (SBS_DBGCR) .......................... 547
    8.5.5 SBS debug lock register (SBS_DBGLOCKR) .......................... 547
    8.5.6 SBS RSS command register (SBS_RSSCMRD) ........................ 548
    8.5.7 SBS product mode and configuration register (SBS_PMCR) ........ 548
    8.5.8 SBS FPU interrupt mask register (SBS_FPUIMR) .................. 549
    8.5.9 SBS memory erase status register (SBS_MESR) ................... 550
    8.5.10 SBS I/O compensation cell control and status register (SBS_CCCSR) .............................. 550
    8.5.11 SBS compensation cell for I/Os value register (SBS_CCVALR) ...... 553
    8.5.12 SBS compensation cell for I/Os software value register (SBS_CCSWVALR) ...................... 554
    8.5.13 SBS break lockup register (SBS_BKLOCKR) ...................... 555
    8.5.14 SBS external interrupt configuration register x (SBS_EXTICRx) .... 557
# Contents

8.5.15 SBS register map .................................................. 558

9 Clock recovery system (CRS) ........................................... 560
  9.1 CRS introduction ..................................................... 560
  9.2 CRS main features .................................................. 560
  9.3 CRS implementation ................................................ 560
  9.4 CRS functional description ....................................... 561
    9.4.1 CRS block diagram ............................................. 561
  9.5 CRS internal signals ................................................ 562
    9.5.1 Synchronization input ......................................... 562
    9.5.2 Frequency error measurement ................................ 562
    9.5.3 Frequency error evaluation and automatic trimming ....... 563
    9.5.4 CRS initialization and configuration ......................... 564
  9.6 CRS in low-power modes ............................................ 565
  9.7 CRS interrupts ..................................................... 565
  9.8 CRS registers ...................................................... 565
    9.8.1 CRS control register (CRS_CR) ................................. 565
    9.8.2 CRS configuration register (CRS_CFGR) ....................... 567
    9.8.3 CRS interrupt and status register (CRS_ISR) ............... 568
    9.8.4 CRS interrupt flag clear register (CRS_ICR) ............... 569
    9.8.5 CRS register map ................................................ 570

10 General-purpose I/Os (GPIO) ........................................ 571
  10.1 Introduction ....................................................... 571
  10.2 GPIO main features ................................................ 571
  10.3 GPIO functional description ..................................... 571
    10.3.1 General-purpose I/O (GPIO) ................................ 573
    10.3.2 I/O pin alternate function multiplexer and mapping .... 574
    10.3.3 I/O port control registers .................................... 574
    10.3.4 I/O port data registers ........................................ 575
    10.3.5 I/O data bitwise handling ..................................... 575
    10.3.6 GPIO locking mechanism ........................................ 575
    10.3.7 I/O alternate function input/output ......................... 575
    10.3.8 External interrupt/wakeup lines ............................... 576
    10.3.9 Input configuration ............................................. 576
    10.3.10 Output configuration .......................................... 577
10.3.11 Alternate function configuration .................................. 577
10.3.12 Analog configuration ............................................... 578
10.3.13 Using the HSE or LSE oscillator pins as GPIOs ............... 578
10.3.14 Using the GPIO pins in the RTC supply domain .............. 579
10.3.15 Privileged and unprivileged modes ............................. 579
10.3.16 High-speed low-voltage mode (HSLV) ......................... 579
10.3.17 I/O compensation cell ........................................... 580

10.4 GPIO registers ......................................................... 580
10.4.1 GPIO port mode register (GPIOx_MODER) 
(x = A to G, M to P) .................................................. 580
10.4.2 GPIO port output type register (GPIOx_OTYPER) 
(x = A to G, M to P) .................................................. 581
10.4.3 GPIO port output speed register (GPIOx_OSPEEDR) 
(x = A to G, M to P) .................................................. 581
10.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) 
(x = A to G, M to P) .................................................. 582
10.4.5 GPIO port input data register (GPIOx_IDR) (x = A to G, M to P) .................................................. 582
10.4.6 GPIO port output data register (GPIOx_ODR) (x = A to G, M to P) .................................................. 583
10.4.7 GPIO port bit set/reset register (GPIOx_BSRR)  
(x = A to G, M to P) .................................................. 583
10.4.8 GPIO port configuration lock register(GPIOx_LCKR) 
(x = A to G, M to P) .................................................. 584
10.4.9 GPIO alternate function low register (GPIOx_AFRL) 
(x = A to G, M to P) .................................................. 585
10.4.10 GPIO alternate function high register 
(GPIOx_AFRH) (x = A to G, M to P) .................................. 586
10.4.11 GPIO port bit reset register (GPIOx_BRR) 
(x = A to G, M to P) .................................................. 587
10.4.12 GPIO register map ................................................ 588

11 Peripheral interconnect .............................................. 590
11.1 Introduction .......................................................... 590
11.2 Connection summary ................................................ 590
11.3 Interconnection details .............................................. 592
11.3.1 Master to slave interconnection for timers .................... 592
11.3.2 Triggers to ADCs .................................................. 593
11.3.3 ADC analog watchdogs as triggers to timers ................. 594
11.3.4 Triggers on ADF1 .................................................. 594
11.3.5 Internal input and clock sources to timers ................... 594
11.3.6 Triggers to low-power timers .................................. 595
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.3.7</td>
<td>RTC wakeup as inputs to timers</td>
<td>596</td>
</tr>
<tr>
<td>11.3.8</td>
<td>USB OTG SOF as trigger to timers</td>
<td>596</td>
</tr>
<tr>
<td>11.3.9</td>
<td>System errors as break signals to timers</td>
<td>596</td>
</tr>
<tr>
<td>11.3.10</td>
<td>Triggers to GPDMA/HPDMA</td>
<td>597</td>
</tr>
<tr>
<td>11.3.11</td>
<td>Internal analog signals to analog peripherals</td>
<td>597</td>
</tr>
<tr>
<td>11.3.12</td>
<td>ADC data filtering by the ADF1</td>
<td>598</td>
</tr>
<tr>
<td>11.3.13</td>
<td>Internal tamper sources</td>
<td>598</td>
</tr>
<tr>
<td>11.3.14</td>
<td>Output from tamper to RTC</td>
<td>599</td>
</tr>
<tr>
<td>11.3.15</td>
<td>Encryption keys to AES/SAES</td>
<td>599</td>
</tr>
</tbody>
</table>

12 General purpose direct memory access controller (GPDMA) .... 600

12.1 GPDMA introduction ........................................ 600
12.2 GPDMA main features ....................................... 600
12.3 GPDMA implementation ....................................... 601
  12.3.1 GPDMA channels ........................................ 601
  12.3.2 GPDMA autonomous mode in low-power modes ............ 602
  12.3.3 GPDMA requests ........................................ 602
  12.3.4 GPDMA block requests ................................... 605
  12.3.5 GPDMA channels with peripheral early termination .... 606
  12.3.6 GPDMA triggers .......................................... 606
12.4 GPDMA functional description .............................. 608
  12.4.1 GPDMA block diagram ................................... 608
  12.4.2 GPDMA channel state and direct programming without any linked-list 609
  12.4.3 GPDMA channel suspend and resume ...................... 610
  12.4.4 GPDMA channel abort and restart ....................... 610
  12.4.5 GPDMA linked-list data structure ..................... 611
  12.4.6 Linked-list item transfer execution ................... 615
  12.4.7 GPDMA channel state and linked-list programming in run-to-completion mode 616
  12.4.8 GPDMA channel state and linked-list programming in link step mode 619
  12.4.9 GPDMA channel state and linked-list programming 626
  12.4.10 GPDMA FIFO-based transfers ........................... 628
  12.4.11 GPDMA transfer request and arbitration ............... 635
  12.4.12 GPDMA triggered transfer ................................ 639
  12.4.13 GPDMA circular buffering with linked-list programming 640
  12.4.14 GPDMA transfer in peripheral flow-control mode .... 642
  12.4.15 GPDMA privileged/unprivileged channel ................ 643
# GPDMA in debug mode

12.5 GPDMA in debug mode ........................................... 646

# GPDMA in low-power modes

12.6 GPDMA in low-power modes .................................... 646

# GPDMA interrupts

12.7 GPDMA interrupts ................................................ 647

# GPDMA registers

12.8 GPDMA registers .................................................. 648

- 12.8.1 GPDMA privileged configuration register (GPDMA_PRIVCFGR) ..... 648
- 12.8.2 GPDMA configuration lock register (GPDMA_RCFGLOCKR) ........ 649
- 12.8.3 GPDMA masked interrupt status register (GPDMA_MISR) ........... 649
- 12.8.4 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) ........................................... 650
- 12.8.5 GPDMA channel x flag clear register (GPDMA_CxFCR) ............ 650
- 12.8.6 GPDMA channel x status register (GPDMA_CxSR) .................. 651
- 12.8.7 GPDMA channel x control register (GPDMA_CxCR) ................ 653
- 12.8.8 GPDMA channel x transfer register 1 (GPDMA_CxTR1) .......... 655
- 12.8.9 GPDMA channel x transfer register 2 (GPDMA_CxTR2) .......... 658
- 12.8.10 GPDMA channel x block register 1 (GPDMA_CxBR1) ............ 662
- 12.8.11 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) .... 663
- 12.8.12 GPDMA channel x source address register (GPDMA_CxSAR) ..... 666
- 12.8.13 GPDMA channel x destination address register (GPDMA_CxDAR) .... 667
- 12.8.14 GPDMA channel x transfer register 3 (GPDMA_CxTR3) .......... 668
- 12.8.15 GPDMA channel x block register 2 (GPDMA_CxBR2) ............ 669
- 12.8.16 GPDMA channel x linked-list address register (GPDMA_CxLLR) ... 670
- 12.8.17 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) ........................................... 672
- 12.8.18 GPDMA register map ......................................... 673

# High-performance direct memory access controller (HPDMA)

13 High-performance direct memory access controller (HPDMA) ........ 675

- 13.1 HPDMA introduction ............................................ 675
- 13.2 HPDMA main features .......................................... 675
- 13.3 HPDMA implementation ......................................... 676

- 13.3.1 HPDMA channels ............................................. 676
- 13.3.2 HPDMA autonomous mode in low-power modes .................... 677
- 13.3.3 HPDMA requests .............................................. 677
- 13.3.4 HPDMA block requests ....................................... 678
- 13.3.5 HPDMA channels with peripheral early termination ............. 678
- 13.3.6 HPDMA triggers .............................................. 678
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>13.4</td>
<td>HPDMA functional description</td>
<td>681</td>
</tr>
<tr>
<td>13.4.1</td>
<td>HPDMA block diagram</td>
<td>681</td>
</tr>
<tr>
<td>13.4.2</td>
<td>HPDMA channel state and direct programming without any linked-list</td>
<td>681</td>
</tr>
<tr>
<td>13.4.3</td>
<td>HPDMA channel suspend and resume</td>
<td>682</td>
</tr>
<tr>
<td>13.4.4</td>
<td>HPDMA channel abort and restart</td>
<td>683</td>
</tr>
<tr>
<td>13.4.5</td>
<td>HPDMA linked-list data structure</td>
<td>684</td>
</tr>
<tr>
<td>13.4.6</td>
<td>Linked-list item transfer execution</td>
<td>687</td>
</tr>
<tr>
<td>13.4.7</td>
<td>HPDMA channel state and linked-list programming in run-to-completion mode</td>
<td>688</td>
</tr>
<tr>
<td>13.4.8</td>
<td>HPDMA channel state and linked-list programming in link step mode</td>
<td>692</td>
</tr>
<tr>
<td>13.4.9</td>
<td>HPDMA channel state and linked-list programming</td>
<td>699</td>
</tr>
<tr>
<td>13.4.10</td>
<td>HPDMA FIFO-based transfers</td>
<td>701</td>
</tr>
<tr>
<td>13.4.11</td>
<td>HPDMA transfer request and arbitration</td>
<td>717</td>
</tr>
<tr>
<td>13.4.12</td>
<td>HPDMA triggered transfer</td>
<td>721</td>
</tr>
<tr>
<td>13.4.13</td>
<td>HPDMA circular buffering with linked-list programming</td>
<td>722</td>
</tr>
<tr>
<td>13.4.14</td>
<td>HPDMA transfer in peripheral flow-control mode</td>
<td>724</td>
</tr>
<tr>
<td>13.4.15</td>
<td>HPDMA privileged/unprivileged channel</td>
<td>725</td>
</tr>
<tr>
<td>13.4.16</td>
<td>HPDMA error management</td>
<td>725</td>
</tr>
<tr>
<td>13.4.17</td>
<td>HPDMA autonomous mode</td>
<td>727</td>
</tr>
<tr>
<td>13.5</td>
<td>HPDMA in debug mode</td>
<td>728</td>
</tr>
<tr>
<td>13.6</td>
<td>HPDMA in low-power modes</td>
<td>728</td>
</tr>
<tr>
<td>13.7</td>
<td>HPDMA interrupts</td>
<td>729</td>
</tr>
<tr>
<td>13.8</td>
<td>HPDMA registers</td>
<td>730</td>
</tr>
<tr>
<td>13.8.1</td>
<td>HPDMA privileged configuration register (HPDMA_PRIVCFGR)</td>
<td>730</td>
</tr>
<tr>
<td>13.8.2</td>
<td>HPDMA configuration lock register (HPDMA_RCFGLOCKR)</td>
<td>731</td>
</tr>
<tr>
<td>13.8.3</td>
<td>HPDMA masked interrupt status register (HPDMA_MISR)</td>
<td>731</td>
</tr>
<tr>
<td>13.8.4</td>
<td>HPDMA channel x linked-list base address register (HPDMA_CxLBAR)</td>
<td>732</td>
</tr>
<tr>
<td>13.8.5</td>
<td>HPDMA channel x flag clear register (HPDMA_CxFCR)</td>
<td>733</td>
</tr>
<tr>
<td>13.8.6</td>
<td>HPDMA channel x status register (HPDMA_CxSR)</td>
<td>734</td>
</tr>
<tr>
<td>13.8.7</td>
<td>HPDMA channel x control register (HPDMA_CxCR)</td>
<td>735</td>
</tr>
<tr>
<td>13.8.8</td>
<td>HPDMA channel x transfer register 1 (HPDMA_CxTR1)</td>
<td>738</td>
</tr>
<tr>
<td>13.8.9</td>
<td>HPDMA channel x transfer register 2 (HPDMA_CxTR2)</td>
<td>741</td>
</tr>
<tr>
<td>13.8.10</td>
<td>HPDMA channel x block register 1 (HPDMA_CxBR1)</td>
<td>745</td>
</tr>
<tr>
<td>13.8.11</td>
<td>HPDMA channel x alternate block register 1 (HPDMA_CxBR1)</td>
<td>746</td>
</tr>
<tr>
<td>13.8.12</td>
<td>HPDMA channel x source address register (HPDMA_CxSAR)</td>
<td>749</td>
</tr>
</tbody>
</table>
13.8.13 HPDMA channel x destination address register (HPDMA_CxDAR) . . 750
13.8.14 HPDMA channel x transfer register 3 (HPDMA_CxTR3) ............... 751
13.8.15 HPDMA channel x block register 2 (HPDMA_CxBR2) ............... 752
13.8.16 HPDMA channel x linked-list address register (HPDMA_CxLLR) . . 753
13.8.17 HPDMA channel x alternate linked-list address register 
(HPDMA_CxLLR) ........................................................................ 755
13.8.18 HPDMA register map ............................................................... 756

14 Chrom-GRC (GFXMMU) ................................................................. 758
14.1 Introduction ............................................................................... 758
14.2 GFXMMU main features ............................................................ 758
14.3 GFXMMU functional and architectural description ...................... 758
  14.3.1 GFXMMU block diagram ...................................................... 758
  14.3.2 GFXMMU internal signals .................................................... 759
  14.3.3 Virtual memory ................................................................ 759
  14.3.4 Packing and unpacking ....................................................... 761
  14.3.5 MMU architecture ............................................................... 762
14.4 GFXMMU interrupts ................................................................. 765
14.5 GFXMMU registers ................................................................. 766
  14.5.1 GFXMMU configuration register (GFXMMU_CR) .................... 766
  14.5.2 GFXMMU status register (GFXMMU_SR) .............................. 768
  14.5.3 GFXMMU flag clear register (GFXMMU_FCR) ....................... 768
  14.5.4 GFXMMU default value register (GFXMMU_DVR) .................. 769
  14.5.5 GFXMMU default alpha register (GFXMMU_DAR) ................. 769
  14.5.6 GFXMMU buffer x configuration register (GFXMMU_BxCR) ...... 770
  14.5.7 GFXMMU LUT entry x low (GFXMMU_LUTxL) ...................... 770
  14.5.8 GFXMMU LUT entry x high (GFXMMU_LUTxH) ..................... 771
  14.5.9 GFXMMU register map ........................................................ 771

15 Chrom-ART Accelerator controller (DMA2D) .............................. 773
15.1 DMA2D introduction ............................................................... 773
15.2 DMA2D main features ............................................................ 773
15.3 DMA2D functional description ................................................. 774
  15.3.1 General description ............................................................ 774
  15.3.2 DMA2D internal signals ..................................................... 775
  15.3.3 DMA2D control ................................................................. 776
  15.3.4 DMA2D foreground and background FIFOs ....................... 776
15.3.5 DMA2D foreground and background PFC ................................. 776
15.3.6 DMA2D foreground and background CLUT interface .................. 778
15.3.7 DMA2D blender ............................................................ 780
15.3.8 DMA2D output PFC ....................................................... 780
15.3.9 DMA2D output FIFO ..................................................... 780
15.3.10 DMA2D output FIFO byte reordering ................................ 781
15.3.11 DMA2D AXI master port timer ....................................... 783
15.3.12 DMA2D transactions .................................................... 783
15.3.13 DMA2D configuration .................................................. 783
15.3.14 YCbCr support ............................................................ 787
15.3.15 DMA2D transfer control (start, suspend, abort, and completion) ... 787
15.3.16 Watermark ................................................................. 788
15.3.17 Error management ....................................................... 788
15.3.18 AXI dead time ............................................................ 788
15.4 DMA2D interrupts ............................................................ 788
15.5 DMA2D registers ............................................................. 789
  15.5.1 DMA2D control register (DMA2D_CR) ................................. 789
  15.5.2 DMA2D interrupt status register (DMA2D_ISR) ....................... 791
  15.5.3 DMA2D interrupt flag clear register (DMA2D_IFCR) ................ 791
  15.5.4 DMA2D foreground memory address register (DMA2D_FGMAR) .... 792
  15.5.5 DMA2D foreground offset register (DMA2D_FGOR) ................... 792
  15.5.6 DMA2D background memory address register (DMA2D_BGMAR) .... 793
  15.5.7 DMA2D background offset register (DMA2D_BGOR) ................. 793
  15.5.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR) ........ 794
  15.5.9 DMA2D foreground color register (DMA2D_FGCOLR) ................ 796
  15.5.10 DMA2D background PFC control register (DMA2D_BGPFCCR) ....... 796
  15.5.11 DMA2D background color register (DMA2D_BGCOLR) ............... 798
  15.5.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR) 798
  15.5.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR) 799
  15.5.14 DMA2D output PFC control register (DMA2D_OPFCCR) .............. 799
  15.5.15 DMA2D output color register (DMA2D_OCOLR) ..................... 800
  15.5.16 DMA2D output color register [alternate] (DMA2D_OCOLR) ........ 801
  15.5.17 DMA2D output color register [alternate] (DMA2D_OCOLR) ........ 801
  15.5.18 DMA2D output color register [alternate] (DMA2D_OCOLR) ........ 802
  15.5.19 DMA2D output memory address register (DMA2D_OMAR) ............ 802
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>15.5.20</td>
<td>DMA2D output offset register (DMA2D_OOR)</td>
<td>803</td>
</tr>
<tr>
<td>15.5.21</td>
<td>DMA2D number of line register (DMA2D_NLR)</td>
<td>803</td>
</tr>
<tr>
<td>15.5.22</td>
<td>DMA2D line watermark register (DMA2D_LWR)</td>
<td>804</td>
</tr>
<tr>
<td>15.5.23</td>
<td>DMA2D AXI master timer configuration register (DMA2D_AMTCR)</td>
<td>804</td>
</tr>
<tr>
<td>15.5.24</td>
<td>DMA2D foreground CLUT (DMA2D_FGCLUTx)</td>
<td>805</td>
</tr>
<tr>
<td>15.5.25</td>
<td>DMA2D background CLUT (DMA2D_BGCLUTx)</td>
<td>805</td>
</tr>
<tr>
<td>15.5.26</td>
<td>DMA2D register map</td>
<td>806</td>
</tr>
</tbody>
</table>

16 Neo-Chrom graphic processor (GPU2D) ................. 808

16.1 GPU2D introduction .................................. 808
16.2 GPU2D main features .................................. 808
16.3 GPU2D general description ............................ 809
  16.3.1 GPU2D block diagram ................................ 809
  16.3.2 GPU2D pins and internal signals .................... 809

17 Texture cache (ICACHE) .................................. 811

17.1 ICACHE introduction .................................. 811
17.2 ICACHE main features .................................. 811
17.3 ICACHE implementation ................................ 812
17.4 ICACHE functional description ......................... 812
  17.4.1 ICACHE block diagram ................................ 813
  17.4.2 ICACHE reset and clocks ............................ 813
  17.4.3 ICACHE TAG memory .................................. 814
  17.4.4 Direct-mapped ICACHE (1-way cache) .................. 815
  17.4.5 ICACHE enable ....................................... 816
  17.4.6 Cacheable and noncacheable traffic ................ 816
  17.4.7 Cacheable accesses .................................. 817
  17.4.8 ICACHE maintenance .................................. 817
  17.4.9 ICACHE performance monitoring ....................... 818
  17.4.10 ICACHE boot ......................................... 818
17.5 ICACHE low-power modes ................................ 818
17.6 ICACHE error management and interrupts .............. 819
17.7 ICACHE registers ...................................... 819
  17.7.1 ICACHE control register (ICACHE_CR) ............... 819
  17.7.2 ICACHE status register (ICACHE_SR) ............... 820
  17.7.3 ICACHE interrupt enable register (ICACHE_IER) ..... 821
### 17.7.4 ICACHE flag clear register (ICACHE_FCR) .................................................. 821
### 17.7.5 ICACHE hit monitor register (ICACHE_HMONR) ........................................... 822
### 17.7.6 ICACHE miss monitor register (ICACHE_MMONR) ....................................... 822
### 17.7.7 ICACHE register map .................................................................................... 822

#### 18 Graphic timer (GFXTIM) ................................................................................. 824

**18.1 Introduction** .................................................................................................. 824
**18.2 GFXTIM main features** .................................................................................. 824
**18.3 GFXTIM functional description** ...................................................................... 824
  - **18.3.1 Block diagram** ....................................................................................... 824
  - **18.3.2 GFXTIM pins and internal signals** ......................................................... 825
  - **18.3.3 Clock generator** .................................................................................... 826
  - **18.3.4 Example of clock generator configuration** ............................................. 828
  - **18.3.5 Absolute timers** .................................................................................... 832
  - **18.3.6 Relative timers** ..................................................................................... 833
  - **18.3.7 Tearing-effect detection** ..................................................................... 834
  - **18.3.8 Event generator** ................................................................................... 834
  - **18.3.9 Watchdog timer** .................................................................................. 835
**18.4 GFXTIM interrupts** ....................................................................................... 836
**18.5 GFXTIM registers** ........................................................................................ 837
  - **18.5.1 GFXTIM configuration register (GFXTIM_CR) ....................................... 837
  - **18.5.2 GFXTIM clock generator configuration register (GFXTIM_CGCR) .... 838
  - **18.5.3 GFXTIM timers configuration register (GFXTIM_TCR) ..................... 840
  - **18.5.4 GFXTIM timers disable register (GFXTIM_TDR) ............................... 841
  - **18.5.5 GFXTIM events control register (GFXTIM_EVCR) ............................ 842
  - **18.5.6 GFXTIM events selection register (GFXTIM_EVSR) ......................... 842
  - **18.5.7 GFXTIM watchdog timer configuration register (GFXTIM_WDGTCR) 844
  - **18.5.8 GFXTIM interrupt status register (GFXTIM_ISR) ............................... 846
  - **18.5.9 GFXTIM interrupt clear register (GFXTIM_ICR) .............................. 847
  - **18.5.10 GFXTIM interrupt enable register (GFXTIM_IER) ......................... 849
  - **18.5.11 GFXTIM timers status register (GFXTIM_TSR) .............................. 851
  - **18.5.12 GFXTIM line clock counter reload register (GFXTIM_LCCRR) ..... 852
  - **18.5.13 GFXTIM frame clock counter reload register (GFXTIM_FCCRR) ... 852
  - **18.5.14 GFXTIM absolute time register (GFXTIM_ATR) ............................. 852
  - **18.5.15 GFXTIM absolute frame counter register (GFXTIM_AFCR) ........... 853
  - **18.5.16 GFXTIM absolute line counter register (GFXTIM_ALCR) ............. 853
18.5.17 GFXTIM absolute frame counter compare 1 register (GFXTIM_AFCC1R) ............................... 854
18.5.18 GFXTIM absolute line counter compare 1 register (GFXTIM_ALCC1R) ............................... 854
18.5.19 GFXTIM absolute line counter compare 2 register (GFXTIM_ALCC2R) ............................... 854
18.5.20 GFXTIM relative frame counter 1 register (GFXTIM_RFC1R) ................................. 855
18.5.21 GFXTIM relative frame counter 1 reload register (GFXTIM_RFC1RR) ............................... 855
18.5.22 GFXTIM relative frame counter 2 register (GFXTIM_RFC2R) ................................. 856
18.5.23 GFXTIM relative frame counter 2 reload register (GFXTIM_RFC2RR) ............................... 856
18.5.24 GFXTIM watchdog counter register (GFXTIM_WDGCR) .................................. 856
18.5.25 GFXTIM watchdog reload register (GFXTIM_WDGRR) .................................. 857
18.5.26 GFXTIM watchdog pre-alarm register (GFXTIM_WDGPAR) .................................. 857
18.5.27 GFXTIM register map .............................................. 857

19 Nested vectored interrupt controller (NVIC) ........................................ 860
  19.1 NVIC features ......................................................... 860
  19.1.1 SysTick calibration value register .................................. 860
  19.1.2 Interrupt and exception vectors .................................. 861

20 Extended interrupt and event controller (EXTI) .................................. 868
  20.1 EXTI main features ....................................................... 868
  20.2 EXTI block diagram ....................................................... 868
  20.2.1 EXTI connections between peripherals and CPU ......................... 869
  20.3 EXTI functional description ............................................... 870
  20.3.1 EXTI configurable event input - CPU wakeup ......................... 871
  20.3.2 EXTI direct event input - CPU wakeup .................................. 872
  20.4 EXTI event input mapping .............................................. 873
  20.5 EXTI functional behavior ............................................. 875
  20.5.1 EXTI CPU interrupt procedure ..................................... 876
  20.5.2 EXTI CPU event procedure ..................................... 876
  20.5.3 EXTI CPU wakeup procedure ..................................... 876
  20.5.4 EXTI software interrupt/event trigger procedure ..................... 876
  20.6 EXTI registers ..................................................... 877
  20.6.1 EXTI rising trigger selection register (EXTI_RTSR1) .................. 877
  20.6.2 EXTI falling trigger selection register (EXTI_FTSR1) .................. 877
20.6.3 EXTI software interrupt event register (EXTI_SWIER1) .......... 878
20.6.4 EXTI rising trigger selection register ( EXTI_RTSR2) .......... 878
20.6.5 EXTI falling trigger selection register ( EXTI_FTSR2) .......... 879
20.6.6 EXTI software interrupt event register (EXTI_SWIER2) .......... 880
20.6.7 EXTI interrupt mask register (EXTI IMR1) .................. 881
20.6.8 EXTI event mask register (EXTI EMR1) .................. 882
20.6.9 EXTI pending register (EXTI_PR1) .................. 882
20.6.10 EXTI interrupt mask register (EXTI IMR2) .................. 883
20.6.11 EXTI event mask register (EXTI EMR2) .................. 883
20.6.12 EXTI pending register (EXTI_PR2) .................. 884
20.6.13 EXTI interrupt mask register (EXTI IMR3) .................. 885
20.6.14 EXTI event mask register (EXTI EMR3) .................. 885
20.6.15 EXTI register map ........................................ 886

21 Cyclic redundancy check calculation unit (CRC) ................. 888
21.1 Introduction ................................................. 888
21.2 CRC main features .......................................... 888
21.3 CRC functional description .................................. 889
   21.3.1 CRC block diagram ................................... 889
   21.3.2 CRC internal signals .................................. 889
   21.3.3 CRC operation .......................................... 889
21.4 CRC registers ................................................ 891
   21.4.1 CRC data register (CRC_DR) .......................... 891
   21.4.2 CRC independent data register (CRC_IDR) ............ 891
   21.4.3 CRC control register (CRC_CR) ....................... 892
   21.4.4 CRC initial value (CRC_INIT) ......................... 893
   21.4.5 CRC polynomial (CRC_POL) ........................... 893
   21.4.6 CRC register map ....................................... 894

22 CORDIC coprocessor (CORDIC) ............................... 895
22.1 CORDIC introduction ........................................ 895
22.2 CORDIC main features ...................................... 895
22.3 CORDIC functional description ............................... 895
   22.3.1 General description ................................... 895
   22.3.2 CORDIC functions ...................................... 895
   22.3.3 Fixed point representation ............................ 902
### 22.3.4 Scaling factor .................................................. 902
### 22.3.5 Precision ......................................................... 903
### 22.3.6 Zero-overhead mode ............................................ 906
### 22.3.7 Polling mode .................................................... 907
### 22.3.8 Interrupt mode ............................................... 908
### 22.3.9 DMA mode ..................................................... 908

### 22.4 CORDIC registers .................................................. 909
  - 22.4.1 CORDIC control/status register (CORDIC_CSR) ................... 909
  - 22.4.2 CORDIC argument register (CORDIC_WDATA) ....................... 911
  - 22.4.3 CORDIC result register (CORDIC_RDATA) .......................... 912
  - 22.4.4 CORDIC register map ........................................... 912

### 23 Flexible memory controller (FMC) ................................. 913
  - 23.1 FMC main features .............................................. 913
  - 23.2 FMC block diagram ............................................. 914
  - 23.3 FMC internal signals ............................................ 916
  - 23.4 AHB interface ................................................... 916
  - 23.5 AXI interface .................................................... 916
    - 23.5.1 Supported memories and transactions .......................... 917
  - 23.6 External device address mapping .................................. 918
    - 23.6.1 NOR/PSRAM address mapping .................................. 919
    - 23.6.2 NAND flash memory address mapping .......................... 919
    - 23.6.3 SDRAM address mapping ...................................... 920
  - 23.7 NOR flash/PSRAM controller ...................................... 924
    - 23.7.1 External memory interface signals ............................ 925
    - 23.7.2 Supported memories and transactions .......................... 927
    - 23.7.3 General timing rules ........................................ 928
    - 23.7.4 NOR flash/PSRAM controller asynchronous transactions .... 929
    - 23.7.5 Synchronous transactions ................................... 948
    - 23.7.6 NOR/PSRAM controller registers ............................. 954
  - 23.8 NAND flash controller ........................................... 963
    - 23.8.1 External memory interface signals ............................ 963
    - 23.8.2 NAND flash supported memories and transactions .......... 964
    - 23.8.3 Timing diagrams for NAND flash memories .................... 965
    - 23.8.4 NAND flash operations ....................................... 966
    - 23.8.5 NAND flash prewait feature ................................ 967
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>23.8.6</td>
<td>Computation of the error correction code (ECC) in NAND flash memory</td>
<td>968</td>
</tr>
<tr>
<td>23.8.7</td>
<td>NAND flash controller registers</td>
<td>969</td>
</tr>
<tr>
<td>23.9</td>
<td>SDRAM controller</td>
<td>975</td>
</tr>
<tr>
<td>23.9.1</td>
<td>SDRAM controller main features</td>
<td>975</td>
</tr>
<tr>
<td>23.9.2</td>
<td>SDRAM External memory interface signals</td>
<td>975</td>
</tr>
<tr>
<td>23.9.3</td>
<td>SDRAM controller functional description</td>
<td>976</td>
</tr>
<tr>
<td>23.9.4</td>
<td>Low-power modes</td>
<td>983</td>
</tr>
<tr>
<td>23.9.5</td>
<td>SDRAM controller registers</td>
<td>986</td>
</tr>
<tr>
<td>23.9.6</td>
<td>FMC register map</td>
<td>992</td>
</tr>
<tr>
<td>24</td>
<td>Extended-SPI interface (XSPI)</td>
<td>995</td>
</tr>
<tr>
<td>24.1</td>
<td>Introduction</td>
<td>995</td>
</tr>
<tr>
<td>24.2</td>
<td>XSPI main features</td>
<td>995</td>
</tr>
<tr>
<td>24.3</td>
<td>XSPI implementation</td>
<td>996</td>
</tr>
<tr>
<td>24.4</td>
<td>XSPI functional description</td>
<td>997</td>
</tr>
<tr>
<td>24.4.1</td>
<td>XSPI block diagram</td>
<td>997</td>
</tr>
<tr>
<td>24.4.2</td>
<td>XSPI pins and internal signals</td>
<td>1001</td>
</tr>
<tr>
<td>24.4.3</td>
<td>Clock constraints</td>
<td>1002</td>
</tr>
<tr>
<td>24.4.4</td>
<td>XSPI interface to memory modes</td>
<td>1002</td>
</tr>
<tr>
<td>24.4.5</td>
<td>XSPI regular-command protocol</td>
<td>1002</td>
</tr>
<tr>
<td>24.4.6</td>
<td>XSPI regular-command protocol signal interface</td>
<td>1006</td>
</tr>
<tr>
<td>24.4.7</td>
<td>HyperBus protocol</td>
<td>1006</td>
</tr>
<tr>
<td>24.4.8</td>
<td>Specific features</td>
<td>1016</td>
</tr>
<tr>
<td>24.4.9</td>
<td>XSPI operating modes introduction</td>
<td>1018</td>
</tr>
<tr>
<td>24.4.10</td>
<td>XSPI indirect mode</td>
<td>1018</td>
</tr>
<tr>
<td>24.4.11</td>
<td>XSPI automatic status-polling mode</td>
<td>1020</td>
</tr>
<tr>
<td>24.4.12</td>
<td>XSPI memory-mapped mode</td>
<td>1020</td>
</tr>
<tr>
<td>24.4.13</td>
<td>XSPI configuration introduction</td>
<td>1021</td>
</tr>
<tr>
<td>24.4.14</td>
<td>XSPI system configuration</td>
<td>1021</td>
</tr>
<tr>
<td>24.4.15</td>
<td>XSPI device configuration</td>
<td>1021</td>
</tr>
<tr>
<td>24.4.16</td>
<td>XSPI regular-command mode configuration</td>
<td>1024</td>
</tr>
<tr>
<td>24.4.17</td>
<td>XSPI HyperBus protocol configuration</td>
<td>1025</td>
</tr>
<tr>
<td>24.4.18</td>
<td>XSPI error management</td>
<td>1027</td>
</tr>
<tr>
<td>24.4.19</td>
<td>XSPI high-speed interface and calibration</td>
<td>1027</td>
</tr>
<tr>
<td>24.4.20</td>
<td>XSPI BUSY and ABORT</td>
<td>1029</td>
</tr>
<tr>
<td>24.4.21</td>
<td>XSPI reconfiguration or deactivation</td>
<td>1029</td>
</tr>
</tbody>
</table>
### 24.4 NCS behavior
- 24.4.22 NCS behavior .................................................. 1029
- 24.4.23 Software-controlled dual-memory support .................. 1031

### 24.5 Address alignment and data number .............................. 1031

### 24.6 XSPI interrupts ................................................. 1033

### 24.7 XSPI registers .................................................. 1033
- 24.7.1 XSPI control register (XSPI_CR) ............................... 1033
- 24.7.2 XSPI device configuration register 1 (XSPI_DCR1) .......... 1036
- 24.7.3 XSPI device configuration register 2 (XSPI_DCR2) .......... 1038
- 24.7.4 XSPI device configuration register 3 (XSPI_DCR3) .......... 1039
- 24.7.5 XSPI device configuration register 4 (XSPI_DCR4) .......... 1039
- 24.7.6 XSPI status register (XSPI_SR) ............................... 1040
- 24.7.7 XSPI flag clear register (XSPI_FCR) ......................... 1041
- 24.7.8 XSPI data length register (XSPI_DLR) ....................... 1041
- 24.7.9 XSPI address register (XSPI_AR) ............................. 1042
- 24.7.10 XSPI data register (XSPI_DR) ............................... 1043
- 24.7.11 XSPI polling status mask register (XSPI_PSMKR) ........ 1043
- 24.7.12 XSPI polling status match register (XSPI_PSMAR) ....... 1044
- 24.7.13 XSPI polling interval register (XSPI_PIR) ................ 1044
- 24.7.14 XSPI communication configuration register (XSPI_CCR) .. 1044
- 24.7.15 XSPI timing configuration register (XSPI_TCR) .......... 1046
- 24.7.16 XSPI instruction register (XSPI_IR) ....................... 1047
- 24.7.17 XSPI alternate bytes register (XSPI_ABR) ................ 1047
- 24.7.18 XSPI low-power timeout register (XSPI_LPTR) .......... 1048
- 24.7.19 XSPI wrap communication configuration register (XSPI_WPCCR) .................................................. 1048
- 24.7.20 XSPI wrap timing configuration register (XSPI_WPTCR) ... 1050
- 24.7.21 XSPI wrap instruction register (XSPI_WPIR) ............. 1051
- 24.7.22 XSPI wrap alternate byte register (XSPI_WPABR) .......... 1051
- 24.7.23 XSPI write communication configuration register (XSPI_WCCR) .................................................. 1051
- 24.7.24 XSPI write timing configuration register (XSPI_WTCR) ... 1053
- 24.7.25 XSPI write instruction register (XSPI_WIR) ............. 1054
- 24.7.26 XSPI write alternate byte register (XSPI_WABR) .......... 1054
- 24.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR) 1055
- 24.7.28 XSPI full-cycle calibration configuration (XSPI_CALFCR) 1055
- 24.7.29 XSPI DLL master calibration configuration (XSPI_CALMR) 1056
- 24.7.30 XSPI DLL slave output calibration configuration (XSPI_CALSOR) .................................................. 1057
25 XSPI I/O manager (XSPIM) ..................................................... 1062
25.1 Introduction ......................................................... 1062
25.2 XSPIM main features ............................................. 1062
25.3 XSPIM implementation ............................................ 1062
25.4 XSPIM functional description ................................. 1062
  25.4.1 XSPIM block diagram ........................................ 1062
  25.4.2 XSPIM input/output pins ................................... 1063
  25.4.3 XSPIM matrix .................................................. 1063
  25.4.4 XSPIM multiplexed mode ................................... 1064
25.5 Use cases description .............................................. 1064
  25.5.1 XSPIs direct octal mode ..................................... 1065
  25.5.2 XSPI direct 16-bit mode ..................................... 1065
  25.5.3 XSPI dual-octal mode ......................................... 1066
  25.5.4 XSPI swapped mode .......................................... 1067
  25.5.5 Two XSPIs multiplexed mode to Port 1 accessing two external memories ................................. 1067
  25.5.6 Two XSPIs multiplexed mode to Port 2 accessing two external memories .................................. 1068
  25.5.7 XSPI1 and XSPI2 drive a single external memory .......... 1069
  25.5.8 A single XSPI drives two external memories ............... 1069
25.6 XSPIM registers ..................................................... 1071
  25.6.1 XSPIM control register (XSPIM_CR) ......................... 1071
  25.6.2 XSPIM register map ........................................... 1072

26 Delay block (DLYB) ....................................................... 1073
26.1 Introduction ......................................................... 1073
26.2 DLYB main features ............................................. 1073
26.3 DLYB functional description ..................................... 1073
  26.3.1 DLYB diagram ................................................ 1073
  26.3.2 DLYB pins and internal signals ............................ 1074
  26.3.3 General description ........................................... 1074
  26.3.4 Delay line length configuration procedure ............... 1075
  26.3.5 Output clock phase configuration procedure ............. 1075
26.4 DLYB registers ..................................................... 1076
27 Analog-to-digital converters (ADC1/2) ................................................. 1078
   27.1 Introduction ................................................................. 1078
   27.2 ADC main features ......................................................... 1078
   27.3 ADC implementation ...................................................... 1080
   27.4 ADC functional description ............................................. 1081
       27.4.1 ADC block diagram .................................................. 1081
       27.4.2 ADC pins and internal signals .................................. 1082
       27.4.3 ADC clocks ............................................................ 1085
       27.4.4 ADC connectivity ..................................................... 1087
       27.4.5 Slave AHB interface ................................................ 1089
       27.4.6 ADC Deep-power-down mode (DEEPPWD) and ADC voltage
               regulator (ADVREGEN) ............................................. 1089
       27.4.7 Single-ended and differential input channels ................. 1090
       27.4.8 Calibration (ADCAL, ADCALDIF, ADC_CALFACT) ................ 1090
       27.4.9 ADC on-off control (ADEN, ADDIS, ADRDY) ...................... 1093
       27.4.10 Constraints when writing the ADC control bits .............. 1094
       27.4.11 Channel selection (SQRx, JSQRx) ................................ 1095
       27.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)  1096
       27.4.13 Single conversion mode (CONT = 0) ............................. 1098
       27.4.14 Continuous conversion mode (CONT = 1) ......................... 1098
       27.4.15 Starting conversions (ADSTART, JADSTART) ................... 1099
       27.4.16 ADC timing ........................................................... 1100
       27.4.17 Stopping an ongoing conversion (ADSTP, JADSTP) ............ 1101
       27.4.18 Conversion on external trigger and trigger polarity
               (EXTSEL, EXTEN, JEXTSEL, JEXTEN) ........................... 1102
       27.4.19 Injected channel management ..................................... 1104
       27.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) ............. 1105
       27.4.21 Queue of context for injected conversions .................... 1106
       27.4.22 Programmable resolution (RES) - fast conversion mode .... 1114
       27.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP) . 1115
       27.4.24 End of conversion sequence (EOS, JEOS) ........................ 1115
       27.4.25 Timing diagrams example (single/continuous modes,
               hardware/software triggers) .................................. 1116
       27.4.26 Data management .................................................... 1118
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>27.4.27</td>
<td>Managing conversions using the ADF</td>
<td>1124</td>
</tr>
<tr>
<td>27.4.28</td>
<td>Dynamic low-power features</td>
<td>1125</td>
</tr>
<tr>
<td>27.4.29</td>
<td>Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)</td>
<td>1130</td>
</tr>
<tr>
<td>27.4.30</td>
<td>Oversampler</td>
<td>1134</td>
</tr>
<tr>
<td>27.4.31</td>
<td>Dual ADC modes</td>
<td>1141</td>
</tr>
<tr>
<td>27.4.32</td>
<td>Temperature sensor</td>
<td>1154</td>
</tr>
<tr>
<td>27.4.33</td>
<td>VBAT supply monitoring</td>
<td>1156</td>
</tr>
<tr>
<td>27.4.34</td>
<td>Monitoring the internal voltage reference</td>
<td>1157</td>
</tr>
<tr>
<td>27.4.35</td>
<td>Monitoring the supply voltage</td>
<td>1158</td>
</tr>
<tr>
<td>27.5</td>
<td>ADC in low-power modes</td>
<td>1159</td>
</tr>
<tr>
<td>27.6</td>
<td>ADC interrupts</td>
<td>1159</td>
</tr>
<tr>
<td>27.7</td>
<td>ADC registers (for each ADC)</td>
<td>1161</td>
</tr>
<tr>
<td>27.7.1</td>
<td>ADC interrupt and status register (ADC_ISR)</td>
<td>1161</td>
</tr>
<tr>
<td>27.7.2</td>
<td>ADC interrupt enable register (ADC_IER)</td>
<td>1163</td>
</tr>
<tr>
<td>27.7.3</td>
<td>ADC control register (ADC_CR)</td>
<td>1165</td>
</tr>
<tr>
<td>27.7.4</td>
<td>ADC configuration register (ADC_CFGR)</td>
<td>1169</td>
</tr>
<tr>
<td>27.7.5</td>
<td>ADC configuration register 2 (ADC_CFRG2)</td>
<td>1173</td>
</tr>
<tr>
<td>27.7.6</td>
<td>ADC sample time register 1 (ADC_SMTPR1)</td>
<td>1176</td>
</tr>
<tr>
<td>27.7.7</td>
<td>ADC sample time register 2 (ADC_SMTPR2)</td>
<td>1176</td>
</tr>
<tr>
<td>27.7.8</td>
<td>ADC watchdog threshold register 1 (ADC_TR1)</td>
<td>1177</td>
</tr>
<tr>
<td>27.7.9</td>
<td>ADC watchdog threshold register 2 (ADC_TR2)</td>
<td>1178</td>
</tr>
<tr>
<td>27.7.10</td>
<td>ADC watchdog threshold register 3 (ADC_TR3)</td>
<td>1179</td>
</tr>
<tr>
<td>27.7.11</td>
<td>ADC regular sequence register 1 (ADC_SQR1)</td>
<td>1179</td>
</tr>
<tr>
<td>27.7.12</td>
<td>ADC regular sequence register 2 (ADC_SQR2)</td>
<td>1180</td>
</tr>
<tr>
<td>27.7.13</td>
<td>ADC regular sequence register 3 (ADC_SQR3)</td>
<td>1181</td>
</tr>
<tr>
<td>27.7.14</td>
<td>ADC regular sequence register 4 (ADC_SQR4)</td>
<td>1182</td>
</tr>
<tr>
<td>27.7.15</td>
<td>ADC regular data register (ADC_DR)</td>
<td>1183</td>
</tr>
<tr>
<td>27.7.16</td>
<td>ADC injected sequence register (ADC_JSQR)</td>
<td>1183</td>
</tr>
<tr>
<td>27.7.17</td>
<td>ADC offset y register (ADC_OFFRy)</td>
<td>1186</td>
</tr>
<tr>
<td>27.7.18</td>
<td>ADC injected channel y data register (ADC_JDRy)</td>
<td>1187</td>
</tr>
<tr>
<td>27.7.19</td>
<td>ADC analog watchdog 2 configuration register (ADC_AWD2CR)</td>
<td>1188</td>
</tr>
<tr>
<td>27.7.20</td>
<td>ADC analog watchdog 3 configuration register (ADC_AWD3CR)</td>
<td>1188</td>
</tr>
<tr>
<td>27.7.21</td>
<td>ADC Differential mode selection register (ADC_DIFSEL)</td>
<td>1189</td>
</tr>
<tr>
<td>27.7.22</td>
<td>ADC calibration factors (ADC_CALFACT)</td>
<td>1190</td>
</tr>
<tr>
<td>27.7.23</td>
<td>ADC option register (ADC_OR)</td>
<td>1190</td>
</tr>
<tr>
<td>27.8</td>
<td>ADC common registers</td>
<td>1191</td>
</tr>
</tbody>
</table>
27.8.1 ADC common status register (ADC_CSR) ................................. 1191
27.8.2 ADC common control register (ADC_CCR) ............................. 1193
27.8.3 ADC common regular data register for dual mode (ADC_CDR) .... 1196
27.8.4 ADC hardware configuration register (ADC_HWCFGR0) ............ 1196
27.8.5 ADC version register (ADC_VERR) ........................................ 1197
27.8.6 ADC identification register (ADC_IPDR) ................................ 1197
27.8.7 ADC size identification register (ADC_SIDR) ......................... 1198

27.9 ADC register map ..................................................................... 1199

28 Digital temperature sensor (DTS) ................................................. 1203

28.1 Introduction .............................................................................. 1203
28.2 DTS main features .................................................................... 1203
28.3 DTS functional description ....................................................... 1204
  28.3.1 DTS block diagram ............................................................. 1204
  28.3.2 DTS internal signals ............................................................ 1204
  28.3.3 DTS block operation ............................................................ 1205
  28.3.4 Operating modes ................................................................. 1205
  28.3.5 Calibration ......................................................................... 1205
  28.3.6 Prescaler ............................................................................ 1205
  28.3.7 Temperature measurement principles ................................... 1206
  28.3.8 Sampling time ..................................................................... 1207
  28.3.9 Quick measurement mode ................................................... 1207
  28.3.10 Trigger input ..................................................................... 1208
  28.3.11 On-off control and ready flag ............................................. 1208
  28.3.12 Temperature measurement sequence .................................. 1209

28.4 DTS low-power modes ............................................................... 1210

28.5 DTS interrupts .......................................................... 1210
  28.5.1 Temperature window comparator .......................................... 1210
  28.5.2 Synchronous interrupt .......................................................... 1210
  28.5.3 Asynchronous wakeup .......................................................... 1210

28.6 DTS registers .......................................................... 1211
  28.6.1 Temperature sensor configuration register 1 (DTS_CFGR1) .... 1211
  28.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1) ........ 1213
  28.6.3 Temperature sensor ramp value register (DTS_RAMPVALR) .... 1213
  28.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1) .. 1214
  28.6.5 Temperature sensor data register (DTS_DR) ................. 1214
28.6.6 Temperature sensor status register (DTS_SR) ........................................... 1215
28.6.7 Temperature sensor interrupt enable register (DTS_ITENR) ...................... 1216
28.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR) ................. 1217
28.6.9 Temperature sensor option register (DTS_OR) ........................................ 1218
28.6.10 DTS register map .......................................................... 1219

29 Voltage reference buffer (VREFBUF) ......................................................... 1220
29.1 Introduction ................................................................. 1220
29.2 VREFBUF implementation .................................................... 1220
29.3 VREFBUF functional description .............................................. 1220
29.4 VREFBUF trimming .......................................................... 1221
29.5 VREFBUF registers .......................................................... 1221
  29.5.1 VREFBUF control and status register (VREFBUF_CSR) ................. 1221
  29.5.2 VREFBUF calibration control register (VREFBUF_CCR) ............... 1222
  29.5.3 VREFBUF register map .................................................. 1223

30 Audio digital filter (ADF) ................................................................. 1224
30.1 Introduction ................................................................. 1224
30.2 ADF main features ........................................................ 1224
30.3 ADF implementation ........................................................ 1225
30.4 ADF functional description .................................................. 1226
  30.4.1 ADF block diagram ...................................................... 1226
  30.4.2 ADF pins and internal signals ........................................ 1226
  30.4.3 Serial input interface (SITF) ........................................... 1227
  30.4.4 ADC slave interface (ADCITF) ....................................... 1231
  30.4.5 Clock generator (CKGEN) .............................................. 1232
  30.4.6 Bitstream matrix (BSMX) .............................................. 1234
  30.4.7 Digital filter processing (DFLT) ....................................... 1235
  30.4.8 Digital filter acquisition modes ...................................... 1245
  30.4.9 Start-up sequence examples ............................................ 1253
  30.4.10 Sound activity detection (SAD) ........................................ 1254
  30.4.11 Data transfer to memory .............................................. 1262
  30.4.12 Autonomous mode ..................................................... 1265
  30.4.13 Register protection ................................................... 1265
  30.5 ADF low-power modes ..................................................... 1266
  30.6 ADF interrupts ............................................................ 1266
30.7 ADF application information ............................................. 1268
   30.7.1 ADF configuration examples for audio capture .......... 1268
   30.7.2 Programming examples ............................................. 1269
   30.7.3 Connection examples ............................................. 1271
   30.7.4 Global frequency response ....................................... 1272
   30.7.5 Total ADF gain .................................................. 1273
   30.7.6 How to compute SAD thresholds ................................. 1276
30.8 ADF registers .......................................................... 1281
   30.8.1 ADF global control register (ADF_GCR) ....................... 1281
   30.8.2 ADF clock generator control register (ADF_CKGCR) ......... 1282
   30.8.3 ADF serial interface control register 0 (ADF_SITF0CR) .... 1284
   30.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR) .. 1285
   30.8.5 ADF digital filter control register 0 (ADF_DFLT0CR) ....... 1286
   30.8.6 ADF digital filter configuration register 0 (ADF_DFLT0CICR) 1288
   30.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR) 1289
   30.8.8 ADF delay control register 0 (ADF_DLY0CR) ................. 1290
   30.8.9 ADF DFLT0 interrupt enable register (ADF_DFLT0IER) ....... 1291
   30.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR) ... 1292
   30.8.11 ADF SAD control register (ADF_SADCR) ...................... 1293
   30.8.12 ADF SAD configuration register (ADF_SADCFSGR) ........... 1295
   30.8.13 ADF SAD sound level register (ADF_SADSDLVR) ............. 1296
   30.8.14 ADF SAD ambient noise level register (ADF_SADANLVR) .... 1297
   30.8.15 ADF digital filter data register 0 (ADF_DFLT0DR) .......... 1297
   30.8.16 ADF register map ............................................... 1297

31 Digital camera interface pixel pipeline (DCMIPP) .................... 1300
   31.1 Introduction ...................................................... 1300
   31.2 DCMIPP main features .............................................. 1302
   31.3 DCMIPP functional description ................................... 1303
      31.3.1 DCMIPP block diagram ....................................... 1303
      31.3.2 DCMIPP pads and internal signals .......................... 1303
      31.3.3 DCMIPP reset and clocks .................................... 1304
      31.3.4 DCMIPP maximum resolution .................................. 1306
      31.3.5 DCMIPP minimum requirements for frame structure .......... 1306
      31.3.6 Description of DCMIPP pixel format support ............... 1306
   31.4 DCMIPP input and flow control ................................... 1307
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>31.4.1 Parallel input interface</td>
<td>1307</td>
</tr>
<tr>
<td>31.4.2 Frame counter</td>
<td>1311</td>
</tr>
<tr>
<td>31.4.3 Frame control</td>
<td>1311</td>
</tr>
<tr>
<td>31.4.4 Pipe deactivation</td>
<td>1315</td>
</tr>
<tr>
<td>31.5 Pipe0 (dump pipe)</td>
<td>1316</td>
</tr>
<tr>
<td>31.5.1 Overview</td>
<td>1316</td>
</tr>
<tr>
<td>31.5.2 Decimation</td>
<td>1316</td>
</tr>
<tr>
<td>31.5.3 Crop/statistics selection/suppression</td>
<td>1317</td>
</tr>
<tr>
<td>31.5.4 Dump counter</td>
<td>1318</td>
</tr>
<tr>
<td>31.5.5 Double buffer mode</td>
<td>1319</td>
</tr>
<tr>
<td>31.5.6 Pixel packing</td>
<td>1319</td>
</tr>
<tr>
<td>31.5.7 Overrun detection</td>
<td>1320</td>
</tr>
<tr>
<td>31.6 Pixel format description</td>
<td>1321</td>
</tr>
<tr>
<td>31.6.1 Parallel interface formats</td>
<td>1321</td>
</tr>
<tr>
<td>31.6.2 Dump pipe formats</td>
<td>1323</td>
</tr>
<tr>
<td>31.6.3 AXI IP-Plug</td>
<td>1324</td>
</tr>
<tr>
<td>31.7 Shadow registers</td>
<td>1326</td>
</tr>
<tr>
<td>31.8 DCMIPP low power modes</td>
<td>1327</td>
</tr>
<tr>
<td>31.9 DCMIPP interrupts</td>
<td>1328</td>
</tr>
<tr>
<td>31.9.1 Free-running DCMIPP</td>
<td>1328</td>
</tr>
<tr>
<td>31.9.2 Interrupts</td>
<td>1328</td>
</tr>
<tr>
<td>31.9.3 Event pins</td>
<td>1330</td>
</tr>
<tr>
<td>31.10 DCMIPP registers</td>
<td>1331</td>
</tr>
<tr>
<td>31.10.1 DCMIPP IP-Plug global register 1 (DCMIPP_IPGR1)</td>
<td>1331</td>
</tr>
<tr>
<td>31.10.2 DCMIPP IP-Plug global register 2 (DCMIPP_IPGR2)</td>
<td>1332</td>
</tr>
<tr>
<td>31.10.3 DCMIPP IP-Plug global register 3 (DCMIPP_IPGR3)</td>
<td>1332</td>
</tr>
<tr>
<td>31.10.4 DCMIPP IP-Plug identification register (DCMIPP_IPGR8)</td>
<td>1332</td>
</tr>
<tr>
<td>31.10.5 DCMIPP IP-Plug Clientx register 1 (DCMIPP_IPCxR1)</td>
<td>1333</td>
</tr>
<tr>
<td>31.10.6 DCMIPP IP-Plug Clientx register 2 (DCMIPP_IPCxR2)</td>
<td>1334</td>
</tr>
<tr>
<td>31.10.7 DCMIPP IP-Plug Clientx register 3 (DCMIPP_IPCxR3)</td>
<td>1334</td>
</tr>
<tr>
<td>31.10.8 DCMIPP parallel interface control register (DCMIPP_PRCR)</td>
<td>1335</td>
</tr>
<tr>
<td>31.10.9 DCMIPP parallel interface embedded synchronization code register (DCMIPP_PRESCR)</td>
<td>1336</td>
</tr>
<tr>
<td>31.10.10 DCMIPP parallel interface embedded synchronization unmask register (DCMIPP_PRESUR)</td>
<td>1337</td>
</tr>
<tr>
<td>31.10.11 DCMIPP parallel interface interrupt enable register (DCMIPP_PRIER)</td>
<td>1338</td>
</tr>
</tbody>
</table>
31.10.12 DCMIPP parallel interface status register (DCMIPP_PRSR) .......... 1338
31.10.13 DCMIPP parallel interface interrupt clear register
(DCMIPP_PRFCR) ................................................................. 1339
31.10.14 DCMIPP common configuration register (DCMIPP_CMCR) .......... 1339
31.10.15 DCMIPP common frame counter register (DCMIPP_CMFRCR) ...... 1340
31.10.16 DCMIPP common interrupt enable register (DCMIPP_CMIER) ...... 1340
31.10.17 DCMIPP common status register 1 (DCMIPP_CMSR1) .......... 1341
31.10.18 DCMIPP common status register 2 (DCMIPP_CMSR2) .......... 1342
31.10.19 DCMIPP common interrupt clear register (DCMIPP_CMFCR) ...... 1344
31.10.20 DCMIPP Pipe0 flow selection configuration register
(DCMIPP_P0FSCR) ................................................................. 1345
31.10.21 DCMIPP Pipe0 flow control configuration register
(DCMIPP_P0FCTCR) ................................................................. 1345
31.10.22 DCMIPP Pipe0 statistic/crop start register (DCMIPP_P0SCSTR) ... 1346
31.10.23 DCMIPP Pipe0 statistic/crop size register (DCMIPP_P0SCSZR) ... 1346
31.10.24 DCMIPP Pipe0 dump counter register (DCMIPP_P0DCCNTR) .... 1347
31.10.25 DCMIPP Pipe0 dump limit register (DCMIPP_P0DCLMTR) .... 1348
31.10.26 DCMIPP Pipe0 pixel packer configuration register
(DCMIPP_P0PPCR) ................................................................. 1348
31.10.27 DCMIPP Pipe0 pixel packer Memory0 address register 1
(DCMIPP_P0PPM0AR1) .............................................................. 1349
31.10.28 DCMIPP Pipe0 pixel packer Memory0 address register 2
(DCMIPP_P0PPM0AR2) .............................................................. 1350
31.10.29 DCMIPP Pipe0 interrupt enable register (DCMIPP_P0IER) ...... 1350
31.10.30 DCMIPP Pipe0 status register (DCMIPP_P0SR) .......... 1351
31.10.31 DCMIPP Pipe0 interrupt clear register (DCMIPP_P0FCR) .... 1352
31.10.32 DCMIPP Pipe0 current flow control configuration register
(DCMIPP_P0CFCTCR) ............................................................. 1353
31.10.33 DCMIPP Pipe0 current statistic/crop start register
(DCMIPP_P0CSCSTR) ............................................................. 1353
31.10.34 DCMIPP Pipe0 current statistic/crop size register
(DCMIPP_P0CSCSZR) ............................................................. 1354
31.10.35 DCMIPP Pipe0 current pixel packer configuration register
(DCMIPP_P0CPPCR) .............................................................. 1355
31.10.36 DCMIPP Pipe0 current pixel packer Memory0 address register 1
(DCMIPP_P0CPPM0AR1) .......................................................... 1356
31.10.37 DCMIPP Pipe0 current pixel packer Memory0 address register 2
(DCMIPP_P0CPPM0AR2) .......................................................... 1356
31.11 DCMIPP register map ...................................................... 1358

32 Parallel synchronous slave interface (PSSI) .......................... 1361
32.1 Introduction ................................................. 1361
32.2 PSSI main features ........................................ 1361
32.3 PSSI functional description ................................. 1361
  32.3.1 PSSI block diagram .................................... 1362
  32.3.2 PSSI pins and internal signals ........................ 1362
  32.3.3 PSSI clock ............................................. 1363
  32.3.4 PSSI data management ................................. 1363
  32.3.5 PSSI optional control signals ........................ 1365
32.4 PSSI interrupts ............................................. 1368
32.5 PSSI registers .............................................. 1368
  32.5.1 PSSI control register (PSSI_CR) ...................... 1368
  32.5.2 PSSI status register (PSSI_SR) ....................... 1370
  32.5.3 PSSI raw interrupt status register (PSSI_RIS) ....... 1371
  32.5.4 PSSI interrupt enable register (PSSI_IER) .......... 1372
  32.5.5 PSSI masked interrupt status register (PSSI_MIS) .. 1372
  32.5.6 PSSI interrupt clear register (PSSI_ICR) .......... 1373
  32.5.7 PSSI data register (PSSI_DR) ....................... 1373
  32.5.8 PSSI register map .................................... 1374

33 LCD-TFT display controller (LTDC) ............................ 1375
33.1 Introduction ................................................. 1375
33.2 LTDC main features ........................................ 1375
33.3 LTDC functional description ................................. 1376
  33.3.1 LTDC block diagram .................................... 1376
  33.3.2 LTDC pins and internal signals ........................ 1376
  33.3.3 LTDC reset and clocks .................................. 1377
33.4 LTDC programmable parameters ............................. 1379
  33.4.1 LTDC global configuration parameters ................. 1379
  33.4.2 Layer programmable parameters ........................ 1381
33.5 LTDC interrupts ............................................. 1385
33.6 LTDC programming procedure ............................... 1386
33.7 LTDC registers .............................................. 1387
  33.7.1 LTDC synchronization size configuration register (LTDC_SSCR) . 1387
  33.7.2 LTDC back porch configuration register (LTDC_BPCR) .... 1387
  33.7.3 LTDC active width configuration register (LTDC_AWCR) .. 1388
  33.7.4 LTDC total width configuration register (LTDC_TWCR) . 1389
33.7.5 LTDC global control register (LTDC_GCR) ........................................ 1389
33.7.6 LTDC shadow reload configuration register (LTDC_SRCR) ................. 1391
33.7.7 LTDC background color configuration register (LTDC_BCCR) .............. 1391
33.7.8 LTDC interrupt enable register (LTDC_IER) ..................................... 1392
33.7.9 LTDC interrupt status register (LTDC_ISR) ..................................... 1393
33.7.10 LTDC interrupt clear register (LTDC_ICR) .................................... 1393
33.7.11 LTDC line interrupt position configuration register (LTDC_LIPCR) .... 1394
33.7.12 LTDC current position status register (LTDC_CPSR) ....................... 1394
33.7.13 LTDC current display status register (LTDC_CDSR) ....................... 1395
33.7.14 LTDC layer x control register (LTDC_LxCR) .................................. 1395
33.7.15 LTDC layer x window horizontal position configuration register
    (LTDC_LxWHPCR) ............................................................................. 1396
33.7.16 LTDC layer x window vertical position configuration register
    (LTDC_LxWVPCR) ........................................................................... 1397
33.7.17 LTDC layer x color keying configuration register
    (LTDC_LxCKCR) ............................................................................ 1398
33.7.18 LTDC layer x pixel format configuration register
    (LTDC_LxFPCR) .............................................................................. 1398
33.7.19 LTDC layer x constant alpha configuration register
    (LTDC_LxCACR) ............................................................................. 1399
33.7.20 LTDC layer x default color configuration register
    (LTDC_LxDCCR) ............................................................................. 1399
33.7.21 LTDC layer x blending factors configuration register
    (LTDC_LxBFCR) ............................................................................. 1400
33.7.22 LTDC layer x color frame buffer address register
    (LTDC_LxCFBAR) ........................................................................... 1401
33.7.23 LTDC layer x color frame buffer length register
    (LTDC_LxCFBLR) ........................................................................... 1401
33.7.24 LTDC layer x color frame buffer line number register
    (LTDC_LxCFBLNR) ........................................................................ 1402
33.7.25 LTDC layer x CLUT write register (LTDC_LxCLUTFWR) .................. 1402
33.7.26 LTDC register map ...................................................................... 1403

34 JPEG codec (JPEG) ............................................................................ 1406
34.1 Introduction .................................................................................. 1406
34.2 JPEG codec main features .......................................................... 1406
34.3 JPEG codec block functional description ....................................... 1407
    34.3.1 General description ............................................................... 1407
    34.3.2 JPEG internal signals ........................................................... 1407
    34.3.3 JPEG decoding procedure ..................................................... 1408
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.3.4 JPEG encoding procedure</td>
<td>1410</td>
</tr>
<tr>
<td>34.4 JPEG codec interrupts</td>
<td>1413</td>
</tr>
<tr>
<td>34.5 JPEG codec registers</td>
<td>1413</td>
</tr>
<tr>
<td>34.5.1 JPEG codec control register (JPEG_CONFR0)</td>
<td>1413</td>
</tr>
<tr>
<td>34.5.2 JPEG codec configuration register 1 (JPEG_CONFR1)</td>
<td>1414</td>
</tr>
<tr>
<td>34.5.3 JPEG codec configuration register 2 (JPEG_CONFR2)</td>
<td>1415</td>
</tr>
<tr>
<td>34.5.4 JPEG codec configuration register 3 (JPEG_CONFR3)</td>
<td>1415</td>
</tr>
<tr>
<td>34.5.5 JPEG codec configuration register x (JPEG_CONFRx)</td>
<td>1416</td>
</tr>
<tr>
<td>34.5.6 JPEG control register (JPEG_CR)</td>
<td>1417</td>
</tr>
<tr>
<td>34.5.7 JPEG status register (JPEG_SR)</td>
<td>1418</td>
</tr>
<tr>
<td>34.5.8 JPEG clear flag register (JPEG_CFR)</td>
<td>1419</td>
</tr>
<tr>
<td>34.5.9 JPEG data input register (JPEG_DIR)</td>
<td>1420</td>
</tr>
<tr>
<td>34.5.10 JPEG data output register (JPEG_DOR)</td>
<td>1420</td>
</tr>
<tr>
<td>34.5.11 JPEG quantization memory x (JPEG_QMEMx_y)</td>
<td>1421</td>
</tr>
<tr>
<td>34.5.12 JPEG Huffman min (JPEG_HUFFMINx_y)</td>
<td>1421</td>
</tr>
<tr>
<td>34.5.13 JPEG Huffman min x (JPEG_HUFFMINx_y)</td>
<td>1422</td>
</tr>
<tr>
<td>34.5.14 JPEG Huffman base (JPEG_HUFFBASEx)</td>
<td>1422</td>
</tr>
<tr>
<td>34.5.15 JPEG Huffman symbol (JPEG_HUFFSYMBx)</td>
<td>1423</td>
</tr>
<tr>
<td>34.5.16 JPEG DHT memory (JPEG_DHTMEMx)</td>
<td>1424</td>
</tr>
<tr>
<td>34.5.17 JPEG Huffman encoder ACx (JPEG_HUFFENC_ACx_y)</td>
<td>1424</td>
</tr>
<tr>
<td>34.5.18 JPEG Huffman encoder DCx (JPEG_HUFFENC_DCx_y)</td>
<td>1425</td>
</tr>
<tr>
<td>34.5.19 JPEG codec register map</td>
<td>1426</td>
</tr>
<tr>
<td>35 True random number generator (RNG)</td>
<td>1428</td>
</tr>
<tr>
<td>35.1 Introduction</td>
<td>1428</td>
</tr>
<tr>
<td>35.2 RNG main features</td>
<td>1428</td>
</tr>
<tr>
<td>35.3 RNG functional description</td>
<td>1429</td>
</tr>
<tr>
<td>35.3.1 RNG block diagram</td>
<td>1429</td>
</tr>
<tr>
<td>35.3.2 RNG internal signals</td>
<td>1429</td>
</tr>
<tr>
<td>35.3.3 Random number generation</td>
<td>1430</td>
</tr>
<tr>
<td>35.3.4 RNG initialization</td>
<td>1433</td>
</tr>
<tr>
<td>35.3.5 RNG operation</td>
<td>1434</td>
</tr>
<tr>
<td>35.3.6 RNG clocking</td>
<td>1435</td>
</tr>
<tr>
<td>35.3.7 Error management</td>
<td>1435</td>
</tr>
<tr>
<td>35.3.8 RNG low-power use</td>
<td>1436</td>
</tr>
<tr>
<td>35.4 RNG interrupts</td>
<td>1437</td>
</tr>
</tbody>
</table>
35.5 RNG processing time .................................................. 1438
35.6 RNG entropy source validation ........................................ 1438
  35.6.1 Introduction ......................................................... 1438
  35.6.2 Validation conditions ............................................. 1439
35.7 RNG registers .......................................................... 1440
  35.7.1 RNG control register (RNG_CR) ................................. 1440
  35.7.2 RNG status register (RNG_SR) ................................... 1442
  35.7.3 RNG data register (RNG_DR) .................................... 1443
  35.7.4 RNG noise source control register (RNG_NSCR) .............. 1443
  35.7.5 RNG health test control register (RNG_HTCR) ................ 1444
  35.7.6 RNG register map ............................................... 1445

36 Secure AES coprocessor (SAES) ........................................ 1446
  36.1 Introduction .......................................................... 1446
  36.2 SAES main features .................................................. 1446
  36.3 SAES implementation ............................................... 1447
  36.4 SAES functional description ........................................ 1447
    36.4.1 SAES block diagram ............................................. 1447
    36.4.2 SAES internal signals .......................................... 1448
    36.4.3 SAES reset and clocks ......................................... 1449
    36.4.4 SAES symmetric cipher implementation ....................... 1449
    36.4.5 SAES encryption or decryption typical usage ................ 1450
    36.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication ..................... 1452
    36.4.7 SAES ciphertext stealing and data padding ................... 1452
    36.4.8 SAES suspend and resume operations ......................... 1453
    36.4.9 SAES basic chaining modes (ECB, CBC) ....................... 1453
    36.4.10 SAES counter (CTR) mode .................................... 1458
    36.4.11 SAES Galois/counter mode (GCM) .............................. 1460
    36.4.12 SAES Galois message authentication code (GMAC) .......... 1464
    36.4.13 SAES counter with CBC-MAC (CCM) ............................. 1466
    36.4.14 SAES operation with wrapped keys ............................ 1471
    36.4.15 SAES operation with shared keys .............................. 1475
    36.4.16 SAES data registers and data swapping ...................... 1476
    36.4.17 SAES key registers ............................................ 1479
    36.4.18 SAES initialization vector registers ....................... 1480
    36.4.19 SAES error management ...................................... 1480
36.5 SAES interrupts ........................................ 1482
36.6 SAES DMA requests .................................... 1483
36.7 SAES processing latency ............................... 1483
36.8 SAES registers .......................................... 1485
   36.8.1 SAES control register (SAES_CR) ................. 1485
   36.8.2 SAES status register (SAES_SR) ................. 1488
   36.8.3 SAES data input register (SAES_DINR) .......... 1489
   36.8.4 SAES data output register (SAES_DOUTR) ..... 1490
   36.8.5 SAES key register 0 (SAES_KEYR0) ........... 1490
   36.8.6 SAES key register 1 (SAES_KEYR1) ........... 1491
   36.8.7 SAES key register 2 (SAES_KEYR2) ........... 1491
   36.8.8 SAES key register 3 (SAES_KEYR3) ........... 1491
   36.8.9 SAES initialization vector register 0 (SAES_IVR0) .... 1492
   36.8.10 SAES initialization vector register 1 (SAES_IVR1) .... 1492
   36.8.11 SAES initialization vector register 2 (SAES_IVR2) .... 1492
   36.8.12 SAES initialization vector register 3 (SAES_IVR3) .... 1493
   36.8.13 SAES key register 4 (SAES_KEYR4) ........... 1493
   36.8.14 SAES key register 5 (SAES_KEYR5) ........... 1493
   36.8.15 SAES key register 6 (SAES_KEYR6) ........... 1494
   36.8.16 SAES key register 7 (SAES_KEYR7) ........... 1494
   36.8.17 SAES suspend registers (SAES_SUSPRx) ....... 1494
   36.8.18 SAES interrupt enable register (SAES_IER) .... 1495
   36.8.19 SAES interrupt status register (SAES_ISR) .... 1496
   36.8.20 SAES interrupt clear register (SAES_ICR) ... 1497
   36.8.21 SAES register map .................................. 1497

37 Cryptographic processor (CRYP) ...................... 1500
   37.1 Introduction ........................................ 1500
   37.2 CRYP main features .................................. 1500
   37.3 CRYP implementation .................................. 1501
   37.4 CRYP functional description ....................... 1502
      37.4.1 CRYP block diagram ........................................ 1502
      37.4.2 CRYP internal signals .............................. 1503
      37.4.3 CRYP reset and clocks .............................. 1503
      37.4.4 CRYP symmetric cipher implementation .......... 1503
      37.4.5 CRYP encryption/decryption typical usage .......... 1504
37.4.6 CRYP authenticated encryption, decryption, and cipher-based message authentication ........................................... 1507
37.4.7 CRYP ciphertext stealing and data padding .......................... 1507
37.4.8 CRYP suspend and resume operations .............................. 1508
37.4.9 CRYP basic chaining modes (ECB, CBC) ............................ 1508
37.4.10 CRYP counter mode (CTR) ........................................ 1513
37.4.11 CRYP AES Galois/counter mode (GCM) ........................... 1515
37.4.12 CRYP AES Galois message authentication code (GMAC) ... 1521
37.4.13 CRYP AES Counter with CBC-MAC (CCM) ...................... 1522
37.4.14 AES key sharing with secure AES co-processor .................. 1527
37.4.15 CRYP data registers and data swapping .......................... 1528
37.4.16 CRYP key registers .................................................. 1531
37.4.17 CRYP initialization vector registers ............................... 1531
37.4.18 CRYP error management ............................................. 1532
37.5 CRYP interrupts ............................................................ 1532
37.6 CRYP DMA requests ...................................................... 1533
37.7 CRYP processing time .................................................... 1534
37.8 CRYP registers ............................................................. 1535
  37.8.1 CRYP control register (CRYP_CR) ................................. 1535
  37.8.2 CRYP status register (CRYP_SR) ................................. 1537
  37.8.3 CRYP data input register (CRYP_DINR) ......................... 1539
  37.8.4 CRYP data output register (CRYP_DOUTR) ...................... 1539
  37.8.5 CRYP DMA control register (CRYP_DMACR) .................... 1540
  37.8.6 CRYP interrupt mask set/clear register (CRYP_IMSCR) ...... 1540
  37.8.7 CRYP raw interrupt status register (CRYP_RISR) ............ 1541
  37.8.8 CRYP masked interrupt status register (CRYP_MISR) ....... 1541
  37.8.9 CRYP key register 0L (CRYP_K0LR) ............................. 1542
  37.8.10 CRYP key register 0R (CRYP_K0RR) ........................... 1543
  37.8.11 CRYP key register 1L (CRYP_K1LR) ........................... 1543
  37.8.12 CRYP key register 1R (CRYP_K1RR) ........................... 1543
  37.8.13 CRYP key register 2L (CRYP_K2LR) ........................... 1544
  37.8.14 CRYP key register 2R (CRYP_K2RR) ........................... 1544
  37.8.15 CRYP key register 3L (CRYP_K3LR) ........................... 1545
  37.8.16 CRYP key register 3R (CRYP_K3RR) ........................... 1545
  37.8.17 CRYP initialization vector register 0L (CRYP_IV0LR) ...... 1545
  37.8.18 CRYP initialization vector register 0R (CRYP_IV0RR) ...... 1546
  37.8.19 CRYP initialization vector register 1L (CRYP_IV1LR) ...... 1546
37.8.20 CRYP initialization vector register 1R (CRYP_IV1RR) .................................. 1547
37.8.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR) 1547
37.8.22 CRYP context swap GCM registers (CRYP_CSGCMxR) .............................. 1548
37.8.23 CRYP register map .................................................................................. 1548

38 Hash processor (HASH) ................................................................. 1550
38.1 Introduction .............................................................................. 1550
38.2 HASH main features .............................................................. 1550
38.3 HASH implementation ............................................................ 1551
38.4 HASH functional description .................................................... 1551
38.4.1 HASH block diagram .............................................................. 1551
38.4.2 HASH internal signals ............................................................. 1551
38.4.3 About secure hash algorithms .................................................. 1552
38.4.4 Message data feeding .............................................................. 1552
38.4.5 Message digest computing ....................................................... 1554
38.4.6 Message padding ..................................................................... 1555
38.4.7 HMAC operation .................................................................... 1557
38.4.8 HASH suspend/resume operations .............................................. 1559
38.4.9 HASH DMA interface ............................................................... 1561
38.4.10 HASH error management ......................................................... 1561
38.4.11 HASH processing time ............................................................ 1561
38.5 HASH interrupts ..................................................................... 1562
38.6 HASH registers ....................................................................... 1562
38.6.1 HASH control register (HASH_CR) .............................................. 1562
38.6.2 HASH data input register (HASH_DIN) ........................................... 1564
38.6.3 HASH start register (HASH_STR) ................................................. 1565
38.6.4 HASH digest registers .............................................................. 1566
38.6.5 HASH interrupt enable register (HASH_IMR) .............................. 1568
38.6.6 HASH status register (HASH_SR) .................................................. 1568
38.6.7 HASH context swap registers ..................................................... 1569
38.6.8 HASH register map ................................................................. 1570

39 Memory cipher engine (MCE) ....................................................... 1572
39.1 Introduction .............................................................................. 1572
39.2 MCE main features ................................................................. 1572
39.3 MCE implementation ............................................................... 1573
39.4  MCE functional description  ...................................................... 1574
  39.4.1  MCE block diagram ......................................................... 1574
  39.4.2  MCE internal signals ...................................................... 1574
  39.4.3  MCE programming .......................................................... 1575
  39.4.4  MCE reset and clocks ..................................................... 1576
  39.4.5  MCE block cipher encryption mode .................................... 1576
  39.4.6  MCE stream cipher encryption mode ................................... 1578
  39.4.7  MCE AXI traffic management ............................................ 1579
  39.4.8  MCE encryption disable options ...................................... 1579
  39.4.9  MCE error management ................................................... 1580
39.5  MCE interrupts ................................................................. 1580
39.6  MCE registers ................................................................. 1581
  39.6.1  MCE configuration register (MCE_CR) ................................ 1581
  39.6.2  MCE status register (MCE_SR) ......................................... 1581
  39.6.3  MCE illegal access status register (MCE_IASR) ..................... 1582
  39.6.4  MCE illegal access clear register (MCE_IACR) ...................... 1583
  39.6.5  MCE illegal access interrupt enable register (MCE_IAIER) ..... 1583
  39.6.6  MCE privileged configuration register (MCE_PRIVCFGR) .......... 1584
  39.6.7  MCE illegal access error status register (MCE_IAESR) .......... 1584
  39.6.8  MCE illegal address register (MCE_IADDR) .......................... 1585
  39.6.9  MCE region x configuration register (MCE_REGCRx) ................ 1585
  39.6.10 MCE start address for region x register (MCE_SADDRx) ........ 1587
       MCE end address for region x register (MCE_EADDRx) ................. 1587
  39.6.11 MCE attribute for region x register (MCE_ATTRx) ................. 1588
  39.6.12 MCE master key x (MCE_MKEYRx) ...................................... 1588
  39.6.13 MCE fast master key x (MCE_FMKEYRx) ................................ 1588
  39.6.14 MCE cipher context z configuration register (MCE_CCzCFGR) ... 1589
  39.6.15 MCE cipher context z nonce register 0 (MCE_CCzNR0) .......... 1591
  39.6.16 MCE cipher context z nonce register 1 (MCE_CCzNR1) .......... 1591
  39.6.17 MCE cipher context z nonce register 2 (MCE_CCzNR2) .......... 1591
  39.6.18 MCE cipher context z key register 0 (MCE_CCzKEYR0) .......... 1592
  39.6.19 MCE cipher context z key register 1 (MCE_CCzKEYR1) .......... 1592
  39.6.20 MCE cipher context z key register 2 (MCE_CCzKEYR2) .......... 1593
  39.6.21 MCE cipher context z key register 3 (MCE_CCzKEYR3) .......... 1593
  39.6.22 MCE register map ........................................................ 1594
40 Public key accelerator (PKA) ........................................ 1596
40.1 Introduction .............................................. 1596
40.2 PKA main features ...................................... 1596
40.3 PKA functional description ............................. 1597
  40.3.1 PKA block diagram .................................. 1597
  40.3.2 PKA internal signals .................................. 1597
  40.3.3 PKA reset and clocks ................................ 1597
  40.3.4 PKA public key acceleration ......................... 1598
  40.3.5 Typical applications for PKA ......................... 1600
  40.3.6 PKA procedure to perform an operation .......... 1602
  40.3.7 PKA error management ............................... 1603
40.4 PKA operating modes ..................................... 1604
  40.4.1 Introduction .......................................... 1604
  40.4.2 Montgomery parameter computation ............... 1605
  40.4.3 Modular addition .................................... 1605
  40.4.4 Modular subtraction .................................. 1606
  40.4.5 Modular and Montgomery multiplication .......... 1606
  40.4.6 Modular exponentiation ............................. 1607
  40.4.7 Modular inversion .................................... 1609
  40.4.8 Modular reduction ................................... 1609
  40.4.9 Arithmetic addition .................................. 1610
  40.4.10 Arithmetic subtraction .............................. 1610
  40.4.11 Arithmetic multiplication ......................... 1611
  40.4.12 Arithmetic comparison ............................. 1611
  40.4.13 RSA CRT exponentiation ............................ 1611
  40.4.14 Point on elliptic curve Fp check ................. 1612
  40.4.15 ECC Fp scalar multiplication ...................... 1613
  40.4.16 ECDSA sign ........................................ 1614
  40.4.17 ECDSA verification ................................. 1616
  40.4.18 ECC complete addition ............................. 1617
  40.4.19 ECC double base ladder ............................ 1617
  40.4.20 ECC projective to affine ........................... 1618
40.5 Example of configurations and processing times .... 1619
  40.5.1 Supported elliptic curves ......................... 1619
  40.5.2 Computation times .................................... 1621
40.6 PKA interrupts ............................................ 1623
40.7 PKA registers .......................................................... 1624
  40.7.1 PKA control register (PKA_CR) .................................. 1624
  40.7.2 PKA status register (PKA_SR) .................................. 1626
  40.7.3 PKA clear flag register (PKA_CLRFR) ......................... 1627
  40.7.4 PKA RAM ......................................................... 1627
  40.7.5 PKA register map ............................................... 1628

41 Advanced-control timers (TIM1) ................................. 1629
  41.1 TIM1 introduction .................................................. 1629
  41.2 TIM1 main features ............................................... 1630
  41.3 TIM1 functional description .................................... 1631
    41.3.1 Block diagram ............................................... 1631
    41.3.2 TIM1 pins and internal signals ............................. 1632
    41.3.3 Time-base unit .............................................. 1636
    41.3.4 Counter modes .............................................. 1638
    41.3.5 Repetition counter ......................................... 1650
    41.3.6 External trigger input ..................................... 1651
    41.3.7 Clock selection ............................................. 1652
    41.3.8 Capture/compare channels .................................. 1656
    41.3.9 Input capture mode .......................................... 1658
    41.3.10 PWM input mode ............................................ 1659
    41.3.11 Forced output mode ........................................ 1660
    41.3.12 Output compare mode ...................................... 1661
    41.3.13 PWM mode .................................................. 1662
    41.3.14 Asymmetric PWM mode ..................................... 1670
    41.3.15 Combined PWM mode ....................................... 1671
    41.3.16 Combined 3-phase PWM mode ................................ 1672
    41.3.17 Complementary outputs and dead-time insertion .......... 1673
    41.3.18 Using the break function ................................ 1676
    41.3.19 Bidirectional break inputs ............................... 1682
    41.3.20 Clearing the tim_ocxref signal on an external event .... 1683
    41.3.21 6-step PWM generation .................................... 1684
    41.3.22 One-pulse mode ............................................ 1685
    41.3.23 Retriggerable One-pulse mode ............................. 1687
    41.3.24 Pulse on compare mode .................................... 1688
    41.3.25 Encoder interface mode .................................... 1690
    41.3.26 Direction bit output ..................................... 1707
### 41.3.27 UIF bit remapping ........................................ 1708
### 41.3.28 Timer input XOR function ................................ 1708
### 41.3.29 Interfacing with Hall sensors .......................... 1708
### 41.3.30 Timer synchronization .................................... 1710
### 41.3.31 ADC triggers .............................................. 1715
### 41.3.32 DMA burst mode ........................................... 1715
### 41.3.33 TIM1 DMA requests ....................................... 1716
### 41.3.34 Debug mode ................................................. 1716
### 41.4 TIM1 low-power modes ....................................... 1717
### 41.5 TIM1 interrupts ............................................ 1717
### 41.6 TIM1 registers ............................................... 1718
   41.6.1 TIM1 control register 1 (TIM1_CR1) ..................... 1718
   41.6.2 TIM1 control register 2 (TIM1_CR2) ..................... 1719
   41.6.3 TIM1 slave mode control register (TIM1_SMCR) .......... 1723
   41.6.4 TIM1 DMA/interrupt enable register (TIM1_DIER) .... 1727
   41.6.5 TIM1 status register (TIM1_SR) ......................... 1728
   41.6.6 TIM1 event generation register (TIM1_EGR) .......... 1731
   41.6.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1) 1732
   41.6.8 TIM1 capture/compare mode register 1 [alternate] (TIM1_CCMR1) ........................................ 1734
   41.6.9 TIM1 capture/compare mode register 2 (TIM1_CCMR2) 1737
   41.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2) ........................................... 1738
   41.6.11 TIM1 capture/compare enable register (TIM1_CCER) .. 1741
   41.6.12 TIM1 counter (TIM1_CNT) ................................. 1745
   41.6.13 TIM1 prescaler (TIM1_PSC) .............................. 1745
   41.6.14 TIM1 autoreload register (TIM1_ARR) .................. 1746
   41.6.15 TIM1 repetition counter register (TIM1_RCR) ....... 1746
   41.6.16 TIM1 capture/compare register 1 (TIM1_CCR1) ....... 1747
   41.6.17 TIM1 capture/compare register 2 (TIM1_CCR2) ....... 1747
   41.6.18 TIM1 capture/compare register 3 (TIM1_CCR3) ....... 1748
   41.6.19 TIM1 capture/compare register 4 (TIM1_CCR4) ....... 1749
   41.6.20 TIM1 break and dead-time register (TIM1_BDTR) ....... 1750
   41.6.21 TIM1 capture/compare register 5 (TIM1_CCR5) ....... 1754
   41.6.22 TIM1 capture/compare register 6 (TIM1_CCR6) ....... 1755
   41.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3) 1756
   41.6.24 TIM1 timer deadtime register 2 (TIM1_DTR2) .......... 1757
41.6.25 TIM1 timer encoder control register (TIM1_ECR) . . . . . . . . . . . . . . . 1758
41.6.26 TIM1 timer input selection register (TIM1_TISEL) . . . . . . . . . . . . . . 1759
41.6.27 TIM1 alternate function option register 1 (TIM1_AF1) . . . . . . . . . . . 1760
41.6.28 TIM1 alternate function register 2 (TIM1_AF2) . . . . . . . . . . . . . . . . . 1763
41.6.29 TIM1 DMA control register (TIM1_DCR) . . . . . . . . . . . . . . . . . . . . . . 1765
41.6.30 TIM1 DMA address for full transfer (TIM1_DMAR) . . . . . . . . . . . . . . 1767
41.6.31 TIM1 register map . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1767

42 General-purpose timers (TIM2/TIM3/TIM4/TIM5) . . . . . . . . . . . . . . . 1770
42.1 TIM2/TIM3/TIM4/TIM5 introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . 1770
42.2 TIM2/TIM3/TIM4/TIM5 main features . . . . . . . . . . . . . . . . . . . . . . . . . . 1770
42.3 TIM2/TIM3/TIM4/TIM5 implementation . . . . . . . . . . . . . . . . . . . . . . . . . 1771
42.4 TIM2/TIM3/TIM4/TIM5 functional description . . . . . . . . . . . . . . . . . . . . 1772
  42.4.1 Block diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1772
  42.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals . . . . . . . . . . . . . . . . 1773
  42.4.3 Time-base unit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1776
  42.4.4 Counter modes . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1778
  42.4.5 Clock selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1790
  42.4.6 Capture/compare channels . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1794
  42.4.7 Input capture mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1796
  42.4.8 PWM input mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1797
  42.4.9 Forced output mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1798
  42.4.10 Output compare mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1798
  42.4.11 PWM mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1800
  42.4.12 Asymmetric PWM mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1808
  42.4.13 Combined PWM mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1809
  42.4.14 Clearing the tim_ocxref signal on an external event . . . . . . . . . . . . . 1810
  42.4.15 One-pulse mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1812
  42.4.16 Retriggerable one-pulse mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1813
  42.4.17 Pulse on compare mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1814
  42.4.18 Encoder interface mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1816
  42.4.19 Direction bit output . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1834
  42.4.20 UIF bit remapping . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1835
  42.4.21 Timer input XOR function . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1835
  42.4.22 Timers and external trigger synchronization . . . . . . . . . . . . . . . . . . . 1835
  42.4.23 Timer synchronization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1839
  42.4.24 ADC triggers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1844
42.4.25 DMA burst mode .................................................. 1845
42.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests ...................... 1846
42.4.27 Debug mode ..................................................... 1846
42.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes .................. 1846
42.4.29 TIM2/TIM3/TIM4/TIM5 interrupts .......................... 1847

42.5 TIM2/TIM3/TIM4/TIM5 registers ................................. 1848
  42.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5) ............ 1848
  42.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5) ............ 1849
  42.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5) . 1851
  42.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5) . 1855
  42.5.5 TIMx status register (TIMx_SR)(x = 2 to 5) ............... 1856
  42.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5) .... 1858
  42.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5) . 1859
  42.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5) . 1861
  42.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5) . 1863
  42.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5) . 1864
  42.5.11 TIMx capture/compare enable register (TIMx_CCE)(x = 2 to 5) . 1867
  42.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5) ..................... 1869
  42.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5) .................... 1869
  42.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 5) .......... 1870
  42.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5) .. 1870
  42.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5) .. 1871
  42.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5) .. 1872
  42.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5) .. 1873
  42.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5) .. 1874
  42.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5) . 1875
  42.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5) . 1876
  42.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5) . 1877
  42.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5) .......... 1878
  42.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5) . 1879
  42.5.25 TIMx register map ............................................ 1881

43 Basic timers (TIM6/TIM7) ............................................ 1884
  43.1 TIM6/TIM7 introduction ......................................... 1884
  43.2 TIM6/TIM7 main features ...................................... 1884
43.3 TIM6/TIM7 functional description .......................... 1884
  43.3.1 TIM6/TIM7 block diagram .......................... 1884
  43.3.2 TIM6/TIM7 internal signals ......................... 1885
  43.3.3 TIM6/TIM7 clocks .................................. 1885
  43.3.4 Time-base unit .................................. 1886
  43.3.5 Counting mode .................................. 1888
  43.3.6 UIF bit remapping ................................ 1894
  43.3.7 ADC triggers .................................. 1895
  43.3.8 TIM6/TIM7 DMA requests .......................... 1895
  43.3.9 Debug mode .................................. 1895
  43.3.10 TIM6/TIM7 low-power modes ....................... 1895
  43.3.11 TIM6/TIM7 interrupts .......................... 1895

43.4 TIM6/TIM7 registers .................................. 1896
  43.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7) .... 1896
  43.4.2 TIMx control register 2 (TIMx_CR2)(x = 6 to 7) .... 1898
  43.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6 to 7) 1898
  43.4.4 TIMx status register (TIMx_SR)(x = 6 to 7) ........ 1899
  43.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7) 1899
  43.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7) ............ 1899
  43.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7) ........... 1900
  43.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7) .... 1900
  43.4.9 TIMx register map ................................ 1901

44 General-purpose timers (TIM9/TIM12/TIM13/TIM14) ........... 1902
  44.1 TIM9/TIM12/TIM13/TIM14 introduction .................. 1902
  44.2 TIM9/TIM12 main features ............................ 1902
  44.3 TIM13/TIM14 main features ............................ 1903
  44.4 TIM9/TIM12/TIM13/TIM14 functional description ...... 1904
    44.4.1 Block diagram .................................. 1904
    44.4.2 TIM9/TIM12/TIM13/TIM14 pins and internal signals 1905
    44.4.3 Time-base unit .................................. 1907
    44.4.4 Counter modes .................................. 1909
    44.4.5 Clock selection .................................. 1912
    44.4.6 Capture/compare channels ......................... 1914
    44.4.7 Input capture mode ............................... 1916
    44.4.8 PWM input mode (TIM9/TIM12 only) ................. 1917
44.4.9 Forced output mode ........................................... 1918
44.4.10 Output compare mode ........................................ 1919
44.4.11 PWM mode .................................................... 1920
44.4.12 Combined PWM mode (TIM9/TIM12 only) ................. 1925
44.4.13 One-pulse mode .............................................. 1926
44.4.14 Retriggerable one pulse mode (TIM9/TIM12 only) ...... 1928
44.4.15 UIF bit remapping ............................................ 1929
44.4.16 Timer input XOR function ................................. 1929
44.4.17 TIM9/TIM12 external trigger synchronization ............ 1929
44.4.18 Slave mode – combined reset + trigger mode .......... 1932
44.4.19 Slave mode – combined reset + gated mode ............ 1932
44.4.20 Timer synchronization (TIM9/TIM12 only) ............... 1933
44.4.21 Using timer output as trigger for other timers
(TIM13/TIM14 only) ............................................... 1933
44.4.22 ADC triggers (TIM9/TIM12 only) ......................... 1934
44.4.23 Debug mode .................................................. 1934

44.5 TIM9/TIM12/TIM13/TIM14 low-power modes ................. 1934

44.6 TIM9/TIM12/TIM13/TIM14 interrupts ......................... 1934

44.7 TIM9/TIM12 registers ........................................... 1935
44.7.1 TIMx control register 1 (TIMx_CR1)(x = 9, 12) .......... 1935
44.7.2 TIM12 control register 2 (TIMx_CR2)(x = 9, 12) ........ 1936
44.7.3 TIMx slave mode control register (TIMx_SMCR)(x = 9, 12) .... 1937
44.7.4 TIMx interrupt enable register (TIMx_DIER)(x = 9, 12) ... 1939
44.7.5 TIMx status register (TIMx_SR)(x = 9, 12) ............. 1940
44.7.6 TIMx event generation register (TIMx_EGR)(x = 9, 12) ... 1941
44.7.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 9, 12) . 1942
44.7.8 TIMx capture/compare mode register 1 [alternate]
(TIMx_CCMR1)(x = 9, 12) ........................................... 1943
44.7.9 TIMx capture/compare enable register (TIMx_CCRER)(x = 9, 12) ... 1946
44.7.10 TIMx counter (TIMx_CNT)(x = 9, 12) .................... 1947
44.7.11 TIMx prescaler (TIMx_PSC)(x = 9, 12) .................. 1948
44.7.12 TIMx autoreload register (TIMx_ARR)(x = 9, 12) ....... 1948
44.7.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 9, 12) ...... 1949
44.7.14 TIMx capture/compare register 2 (TIMx_CCR2)(x = 9, 12) .... 1949
44.7.15 TIMx timer input selection register (TIMx_TISEL)(x = 9, 12) ... 1950
44.7.16 TIM9/TIM12 register map .................................. 1951

44.8 TIM13/TIM14 registers ........................................... 1953
44.8.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14) .......................... 1953
44.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 13, 14) ............. 1954
44.8.3 TIMx status register (TIMx_SR)(x = 13, 14) .......................... 1954
44.8.4 TIMx event generation register (TIMx_EGR)(x = 13, 14) ............. 1955
44.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 13, 14) .......... 1956
44.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 13, 14) .......... 1957
44.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 13, 14) ................. 1959
44.8.8 TIMx counter (TIMx_CNT)(x = 13, 14) ........................................ 1960
44.8.9 TIMx prescaler (TIMx_PSC)(x = 13, 14) .................................... 1961
44.8.10 TIMx autoreload register (TIMx_ARR)(x = 13, 14) ..................... 1961
44.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14) .......... 1962
44.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 13, 14) .......... 1962
44.8.13 TIM13/TIM14 register map ....................................................... 1963

45 General purpose timers (TIM15/TIM16/TIM17) ................................. 1965
45.1 TIM15/TIM16/TIM17 introduction .................................................. 1965
45.2 TIM15 main features ................................................................. 1965
45.3 TIM16/TIM17 main features ......................................................... 1966
45.4 TIM15/TIM16/TIM17 functional description .................................... 1967
45.4.1 Block diagram ........................................................................ 1967
45.4.2 TIM15/TIM16/TIM17 pins and internal signals ............................. 1968
45.4.3 Time-base unit ..................................................................... 1971
45.4.4 Counter modes ..................................................................... 1973
45.4.5 Repetition counter ................................................................ 1977
45.4.6 Clock selection .................................................................... 1978
45.4.7 Capture/compare channels ....................................................... 1980
45.4.8 Input capture mode ............................................................... 1982
45.4.9 PWM input mode (only for TIM15) ......................................... 1984
45.4.10 Forced output mode ............................................................. 1985
45.4.11 Output compare mode ......................................................... 1985
45.4.12 PWM mode ......................................................................... 1987
45.4.13 Combined PWM mode (TIM15 only) ..................................... 1992
45.4.14 Complementary outputs and dead-time insertion .................... 1993
45.4.15 Using the break function ...................................................... 1996
45.4.16 Bidirectional break input ...................................................... 2000
45.4.17 Clearing the tim_ocxref signal on an external event ........................................ 2001
45.4.18 6-step PWM generation ................................................................. 2002
45.4.19 One-pulse mode .............................................................................. 2004
45.4.20 Retriggerable one pulse mode (TIM15 only) ........................................... 2005
45.4.21 UIF bit remapping ............................................................................. 2006
45.4.22 Timer input XOR function (TIM15 only) ............................................... 2006
45.4.23 External trigger synchronization (TIM15 only) ...................................... 2006
45.4.24 Slave mode – combined reset + trigger mode (TIM15 only) .................. 2009
45.4.25 Slave mode – combined reset + gated mode (TIM15 only) ..................... 2009
45.4.26 Timer synchronization (TIM15 only) ................................................. 2010
45.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only) .......... 2010
45.4.28 ADC triggers (TIM15 only) ............................................................... 2010
45.4.29 DMA burst mode ............................................................................... 2010
45.4.30 TIM15/TIM16/TIM17 DMA requests .................................................... 2011
45.4.31 Debug mode ....................................................................................... 2012

45.5 TIM15/TIM16/TIM17 low-power modes ....................................................... 2012

45.6 TIM15/TIM16/TIM17 interrupts ............................................................... 2012

45.7 TIM15 registers ....................................................................................... 2013

45.7.1 TIM15 control register 1 (TIM15_CR1) ................................................. 2013
45.7.2 TIM15 control register 2 (TIM15_CR2) ............................................... 2014
45.7.3 TIM15 slave mode control register (TIM15_SMCR) .............................. 2016
45.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER) ......................... 2018
45.7.5 TIM15 status register (TIM15_SR) ...................................................... 2019
45.7.6 TIM15 event generation register (TIM15_EGR) .................................... 2021
45.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) ................. 2022
45.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) .... 2024
45.7.9 TIM15 capture/compare enable register (TIM15_CCER) ...................... 2026
45.7.10 TIM15 counter (TIM15_CNT) .......................................................... 2029
45.7.11 TIM15 prescaler (TIM15_PSC) .......................................................... 2029
45.7.12 TIM15 autoreload register (TIM15_ARR) ........................................... 2030
45.7.13 TIM15 repetition counter register (TIM15_RCR) ................................. 2030
45.7.14 TIM15 capture/compare register 1 (TIM15_CCR1) ............................ 2031
45.7.15 TIM15 capture/compare register 2 (TIM15_CCR2) ............................ 2032
45.7.16 TIM15 break and dead-time register (TIM15_BDTR) ......................... 2032
45.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2) .............................. 2035
45.7.18 TIM15 input selection register (TIM15_TISEL) .................................... 2036
45.7.19 TIM15 alternate function register 1 (TIM15_AF1) ........................................ 2037
45.7.20 TIM15 alternate function register 2 (TIM15_AF2) ........................................ 2039
45.7.21 TIM15 DMA control register (TIM15_DCR) .................................................. 2040
45.7.22 TIM15 DMA address for full transfer (TIM15_DMAR) .................................... 2041
45.7.23 TIM15 register map .................................................................................... 2041
45.8 TIM16/TIM17 registers .................................................................................... 2044
  45.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) ........................................ 2044
  45.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) ........................................ 2045
  45.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) ................. 2046
  45.8.4 TIMx status register (TIMx_SR)(x = 16 to 17) ............................................. 2047
  45.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) ......................... 2048
  45.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1)
                (x = 16 to 17) .................................................................................... 2049
  45.8.7 TIMx capture/compare mode register 1 [alternate]
                (TIMx_CCMR1)(x = 16 to 17) ................................................................ 2050
  45.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) ............ 2052
  45.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17) ..................................................... 2055
  45.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) .................................................. 2055
  45.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17) ..................... 2056
  45.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) ..................... 2056
  45.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) ................... 2057
  45.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) ............... 2058
  45.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) ..................... 2061
  45.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) ..................... 2062
  45.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) .................... 2062
  45.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) .................... 2065
  45.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) ............................ 2065
  45.8.20 TIM16/TIM17 DMA address for full transfer
                (TIMx_DMAR)(x = 16 to 17) ................................................................ 2066
  45.8.21 TIM16/TIM17 register map .................................................................... 2068

46 Low-power timer (LPTIM) .................................................................................... 2070
  46.1 Introduction .................................................................................................... 2070
  46.2 LPTIM main features .................................................................................... 2070
  46.3 LPTIM implementation .................................................................................. 2071
  46.4 LPTIM functional description ........................................................................ 2072
    46.4.1 LPTIM block diagram .............................................................................. 2072
46.4.2 LPTIM pins and internal signals ........................................... 2073
46.4.3 LPTIM input and trigger mapping ........................................... 2075
46.4.4 LPTIM reset and clocks ....................................................... 2076
46.4.5 Glitch filter ................................................................. 2077
46.4.6 Prescaler ................................................................. 2077
46.4.7 Trigger multiplexer ......................................................... 2078
46.4.8 Operating mode ............................................................. 2078
46.4.9 Timeout function ............................................................ 2080
46.4.10 Waveform generation ......................................................... 2080
46.4.11 Register update ............................................................ 2082
46.4.12 Counter mode ............................................................... 2082
46.4.13 Timer enable ............................................................... 2083
46.4.14 Timer counter reset ........................................................ 2083
46.4.15 Encoder mode ............................................................... 2084
46.4.16 Repetition Counter ......................................................... 2085
46.4.17 Capture/compare channels ................................................ 2087
46.4.18 Input capture mode ........................................................ 2087
46.4.19 PWM mode ................................................................. 2089
46.4.20 DMA requests .............................................................. 2091
46.4.21 Debug mode ............................................................... 2092
46.5 LPTIM low-power modes ....................................................... 2092
46.6 LPTIM interrupts ................................................................. 2092
46.7 LPTIM registers ................................................................. 2093
  46.7.1 LPTIMx interrupt and status register (LPTIMx_ISR)(x = 4 to 5) .... 2094
  46.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR)
      (x = 1 to 3) ................................................................. 2095
  46.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR)
      (x = 1 to 3) ................................................................. 2097
  46.7.4 LPTIMx interrupt clear register (LPTIMx_ICR)(x = 4 to 5) .... 2099
  46.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)
      (x = 1 to 3) ................................................................. 2100
  46.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)
      (x = 1 to 3) ................................................................. 2101
  46.7.7 LPTIMx interrupt enable register (LPTIMx_DIER)(x = 4 to 5) .... 2102
  46.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER)
      (x = 1 to 3) ................................................................. 2104
  46.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER)
      (x = 1 to 3) ................................................................. 2105
<table>
<thead>
<tr>
<th>Section Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>46.7.10</td>
<td>LPTIM configuration register (LPTIM_CFGR)</td>
<td>2107</td>
</tr>
<tr>
<td>46.7.11</td>
<td>LPTIM control register (LPTIM_CR)</td>
<td>2110</td>
</tr>
<tr>
<td>46.7.12</td>
<td>LPTIM compare register 1 (LPTIM_CCR1)</td>
<td>2111</td>
</tr>
<tr>
<td>46.7.13</td>
<td>LPTIM autoreload register (LPTIM_ARR)</td>
<td>2112</td>
</tr>
<tr>
<td>46.7.14</td>
<td>LPTIM counter register (LPTIM_CNT)</td>
<td>2112</td>
</tr>
<tr>
<td>46.7.15</td>
<td>LPTIM configuration register 2 (LPTIM_CFG2)</td>
<td>2113</td>
</tr>
<tr>
<td>46.7.16</td>
<td>LPTIM repetition register (LPTIM_RCR)</td>
<td>2114</td>
</tr>
<tr>
<td>46.7.17</td>
<td>LPTIM capture/compare mode register 1 (LPTIM_CCMR1)</td>
<td>2114</td>
</tr>
<tr>
<td>46.7.18</td>
<td>LPTIM compare register 2 (LPTIM_CCR2)</td>
<td>2117</td>
</tr>
<tr>
<td>46.7.19</td>
<td>LPTIM register map</td>
<td>2117</td>
</tr>
<tr>
<td>47</td>
<td><strong>System window watchdog (WWDG)</strong></td>
<td>2120</td>
</tr>
<tr>
<td>47.1</td>
<td>Introduction</td>
<td>2120</td>
</tr>
<tr>
<td>47.2</td>
<td>WWGD main features</td>
<td>2120</td>
</tr>
<tr>
<td>47.3</td>
<td>WWGD implementation</td>
<td>2120</td>
</tr>
<tr>
<td>47.4</td>
<td>WWGD functional description</td>
<td>2121</td>
</tr>
<tr>
<td>47.4.1</td>
<td>WWGD block diagram</td>
<td>2121</td>
</tr>
<tr>
<td>47.4.2</td>
<td>WWGD internal signals</td>
<td>2121</td>
</tr>
<tr>
<td>47.4.3</td>
<td>Enabling the watchdog</td>
<td>2122</td>
</tr>
<tr>
<td>47.4.4</td>
<td>Controlling the down-counter</td>
<td>2122</td>
</tr>
<tr>
<td>47.4.5</td>
<td>How to program the watchdog timeout</td>
<td>2122</td>
</tr>
<tr>
<td>47.4.6</td>
<td>Debug mode</td>
<td>2123</td>
</tr>
<tr>
<td>47.5</td>
<td>WWGD interrupts</td>
<td>2124</td>
</tr>
<tr>
<td>47.6</td>
<td>WWGD registers</td>
<td>2124</td>
</tr>
<tr>
<td>47.6.1</td>
<td>WWGD control register (WWGD_CR)</td>
<td>2124</td>
</tr>
<tr>
<td>47.6.2</td>
<td>WWGD configuration register (WWGD_CFG)</td>
<td>2125</td>
</tr>
<tr>
<td>47.6.3</td>
<td>WWGD status register (WWGD_SR)</td>
<td>2125</td>
</tr>
<tr>
<td>47.6.4</td>
<td>WWGD register map</td>
<td>2126</td>
</tr>
<tr>
<td>48</td>
<td><strong>Independent watchdog (IWDG)</strong></td>
<td>2127</td>
</tr>
<tr>
<td>48.1</td>
<td>Introduction</td>
<td>2127</td>
</tr>
<tr>
<td>48.2</td>
<td>IWDG main features</td>
<td>2127</td>
</tr>
<tr>
<td>48.3</td>
<td>IWDG implementation</td>
<td>2127</td>
</tr>
<tr>
<td>48.4</td>
<td>IWDG functional description</td>
<td>2128</td>
</tr>
<tr>
<td>48.4.1</td>
<td>IWDG block diagram</td>
<td>2128</td>
</tr>
<tr>
<td>48.4.2</td>
<td>IWDG internal signals</td>
<td>2129</td>
</tr>
</tbody>
</table>
## 48 IWDG

48.3 Software and hardware watchdog modes ........................................ 2129
48.4 Window option ............................................................................. 2130
48.5 Debug ............................................................................................ 2133
48.6 Register access protection ............................................................... 2133

### 48.5 IWDG low power modes ............................................................ 2133

### 48.6 IWDG interrupts ....................................................................... 2134

### 48.7 IWDG registers ......................................................................... 2136

#### 48.7.1 IWDG key register (IWDG_KR) ............................................. 2137
#### 48.7.2 IWDG prescaler register (IWDG_PR) ..................................... 2137
#### 48.7.3 IWDG reload register (IWDG_RLR) ...................................... 2138
#### 48.7.4 IWDG status register (IWDG_SR) ......................................... 2138
#### 48.7.5 IWDG window register (IWDG_WINR) ................................. 2140
#### 48.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) ....... 2140
#### 48.7.7 IWDG register map ............................................................... 2142

## 49 Real-time clock (RTC) ................................................................. 2143

49.1 Introduction .................................................................................... 2143
49.2 RTC main features ......................................................................... 2143
49.3 RTC functional description ............................................................ 2143

#### 49.3.1 RTC block diagram ............................................................... 2143
#### 49.3.2 RTC pins and internal signals .............................................. 2145
#### 49.3.3 GPIOs controlled by the RTC and TAMP .......................... 2146
#### 49.3.4 RTC privilege protection modes .......................................... 2149
#### 49.3.5 Clock and prescalers ............................................................. 2150
#### 49.3.6 Real-time clock and calendar ............................................. 2151
#### 49.3.7 Calendar ultra-low power mode ......................................... 2152
#### 49.3.8 Programmable alarms .......................................................... 2152
#### 49.3.9 Periodic auto-wake-up ......................................................... 2152
#### 49.3.10 RTC initialization and configuration ................................... 2153
#### 49.3.11 Reading the calendar .......................................................... 2156
#### 49.3.12 Resetting the RTC .............................................................. 2157
#### 49.3.13 RTC synchronization .......................................................... 2157
#### 49.3.14 RTC reference clock detection ........................................... 2158
#### 49.3.15 RTC smooth digital calibration ........................................... 2159
#### 49.3.16 Timestamp function ............................................................ 2161
#### 49.3.17 Calibration clock output ..................................................... 2161
49.3.18 Tamper and alarm output ............................................ 2162
49.4 RTC low-power modes .................................................. 2162
49.5 RTC interrupts ........................................................... 2163
49.6 RTC registers ............................................................ 2163
  49.6.1 RTC time register (RTC_TR) ........................................ 2164
  49.6.2 RTC date register (RTC_DR) ....................................... 2165
  49.6.3 RTC subsecond register (RTC_SSR) ............................... 2166
  49.6.4 RTC initialization control and status register (RTC_ICSR) ..... 2166
  49.6.5 RTC prescaler register (RTC_PRER) .............................. 2168
  49.6.6 RTC wake-up timer register (RTC_WUTR) ....................... 2169
  49.6.7 RTC control register (RTC_CR) ................................... 2169
  49.6.8 RTC privilege mode control register (RTC_PRIVCFGR) ........ 2173
  49.6.9 RTC write protection register (RTC_WPR) ....................... 2175
  49.6.10 RTC calibration register (RTC_CALR) .......................... 2175
  49.6.11 RTC shift control register (RTC_SHIFTR) .................... 2177
  49.6.12 RTC timestamp time register (RTC_TSTR) .................... 2178
  49.6.13 RTC timestamp date register (RTC_TSDR) .................... 2179
  49.6.14 RTC timestamp subsecond register (RTC_TSSSR) ............. 2179
  49.6.15 RTC alarm A register (RTC_ALRMAR) ........................... 2180
  49.6.16 RTC alarm A subsecond register (RTC_ALRMASSR) ........... 2181
  49.6.17 RTC alarm B register (RTC_ALRMBR) ........................... 2182
  49.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR) ........... 2183
  49.6.19 RTC status register (RTC_SR) .................................. 2184
  49.6.20 RTC masked interrupt status register (RTC_MISR) .......... 2186
  49.6.21 RTC status clear register (RTC_SCR) ......................... 2187
  49.6.22 RTC alarm A binary mode register (RTC_ALRABINR) .......... 2188
  49.6.23 RTC alarm B binary mode register (RTC_ALRBBINR) .......... 2188
  49.6.24 RTC register map ............................................... 2190

50 Tamper and backup registers (TAMP) ........................................... 2192
  50.1 Introduction ................................................................... 2192
  50.2 TAMP main features ..................................................... 2192
  50.3 TAMP functional description .......................................... 2193
    50.3.1 TAMP block diagram ............................................. 2193
    50.3.2 TAMP pins and internal signals ................................ 2194
    50.3.3 GPIOs controlled by the RTC and TAMP ...................... 2196
50.3.4 TAMP register write protection ................................................. 2196
50.3.5 Backup registers protection zones ............................................. 2197
50.3.6 TAMP privilege protection modes .............................................. 2197
50.3.7 Boot hardware key (BHK) ......................................................... 2197
50.3.8 Tamper detection ................................................................. 2198
50.3.9 TAMP backup registers and other device secrets erase .................... 2198
50.3.10 Tamper detection configuration and initialization ......................... 2200
50.4 TAMP low-power modes ............................................................ 2205
50.5 TAMP interrupts ................................................................. 2206
50.6 TAMP registers ................................................................. 2206
50.6.1 TAMP control register 1 (TAMP_CR1) .................................... 2206
50.6.2 TAMP control register 2 (TAMP_CR2) .................................... 2208
50.6.3 TAMP control register 3 (TAMP_CR3) .................................... 2210
50.6.4 TAMP filter control register (TAMP_FLTCR) ............................ 2212
50.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) ................ 2213
50.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) ................ 2216
50.6.7 TAMP active tamper output register (TAMP_ATOR) .................... 2216
50.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) ................ 2217
50.6.9 TAMP configuration register (TAMP_CFGR) ............................ 2220
50.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) ........ 2221
50.6.11 TAMP interrupt enable register (TAMP_IER) .......................... 2222
50.6.12 TAMP status register (TAMP_SR) ....................................... 2224
50.6.13 TAMP masked interrupt status register (TAMP_MISR) .............. 2226
50.6.14 TAMP status clear register (TAMP_SCR) ............................... 2228
50.6.15 TAMP monotonic counter 1 register (TAMP_COUNT1R) ............. 2230
50.6.16 TAMP resources protection configuration register (TAMP_RPCFGR) 2230
50.6.17 TAMP backup x register (TAMP_BKPxR) ................................ 2231
50.6.18 TAMP register map .......................................................... 2232

51 Inter-integrated circuit interface (I2C) ........................................ 2234
51.1 Introduction ................................................................................. 2234
51.2 I2C main features ....................................................................... 2234
51.3 I2C implementation ..................................................................... 2235
51.4 I2C functional description .......................................................... 2235
51.4.1 I2C block diagram .................................................................. 2236
51.4.2 I2C pins and internal signals .................................................... 2236
51.4.3 I2C clock requirements ........................................... 2237
51.4.4 Mode selection .................................................... 2237
51.4.5 I2C initialization ................................................... 2238
51.4.6 I2C reset ............................................................ 2242
51.4.7 Data transfer ....................................................... 2243
51.4.8 I2C slave mode ..................................................... 2245
51.4.9 I2C master mode .................................................... 2254
51.4.10 I2C_TIMINGR register configuration examples ............... 2265
51.4.11 SMBus specific features .......................................... 2267
51.4.12 SMBus initialization .............................................. 2269
51.4.13 SMBus I2C_TIMEOUTR register configuration examples ...... 2271
51.4.14 SMBus slave mode ............................................... 2271
51.4.15 SMBus master mode .............................................. 2275
51.4.16 Wake-up from Stop mode on address match .................... 2278
51.4.17 Error conditions .................................................. 2279
51.5 I2C in low-power modes ............................................ 2281
51.6 I2C interrupts ......................................................... 2281
51.7 I2C DMA requests .................................................... 2282
51.8 I2C debug modes ..................................................... 2282
51.9 I2C registers .......................................................... 2283
  51.9.1 I2C control register 1 (I2C_CR1) ................................ 2283
  51.9.2 I2C control register 2 (I2C_CR2) ................................ 2286
  51.9.3 I2C own address 1 register (I2C_OAR1) ......................... 2288
  51.9.4 I2C own address 2 register (I2C_OAR2) ......................... 2288
  51.9.5 I2C timing register (I2C_TIMINGR) ............................ 2289
  51.9.6 I2C timeout register (I2C_TIMEOUTR) ......................... 2290
  51.9.7 I2C interrupt and status register (I2C_ISR) ................... 2291
  51.9.8 I2C interrupt clear register (I2C_ICR) ......................... 2294
  51.9.9 I2C PEC register (I2C_PECR) .................................. 2295
  51.9.10 I2C receive data register (I2C_RXDR) ......................... 2295
  51.9.11 I2C transmit data register (I2C_TXDR) ......................... 2296
  51.9.12 I2C register map ................................................ 2297

52 Improved inter-integrated circuit (I3C) .......................... 2298
  52.1 Introduction ....................................................... 2298
  52.2 I3C main features ................................................ 2298
52.3 I3C implementation ........................................... 2300
  52.3.1 I3C instantiation ........................................ 2300
  52.3.2 I3C wake-up from low-power mode(s) .................. 2300
  52.3.3 I3C FIFOs ................................................. 2300
  52.3.4 I3C triggers ............................................. 2300
  52.3.5 I3C interrupt(s) ......................................... 2300
  52.3.6 I3C MIPI® support ....................................... 2301
52.4 I3C block diagram ............................................ 2302
52.5 I3C pins and internal signals ................................. 2302
52.6 I3C reset and clocks .......................................... 2303
  52.6.1 I3C reset .................................................. 2303
  52.6.2 I3C clocks and requirements ............................... 2303
52.7 I3C peripheral state and programming ......................... 2305
  52.7.1 I3C peripheral state ...................................... 2305
  52.7.2 I3C controller state and programming sequence ......... 2305
  52.7.3 I3C target state and programming sequence ............. 2310
52.8 I3C registers and programming .................................. 2314
  52.8.1 I3C register set, as controller/target .................. 2314
  52.8.2 I3C registers and fields use vs. peripheral state, as controller ..... 2315
  52.8.3 I3C registers and fields usage vs. peripheral state, as target .... 2318
52.9 I3C bus transfers and programming ............................. 2320
  52.9.1 I3C command set (CCCs), as controller/target ........... 2320
  52.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller ...... 2324
  52.9.3 I3C broadcast ENTDAA CCC transfer, as controller ........ 2326
  52.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller ...... 2326
  52.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target ...... 2328
  52.9.6 I3C broadcast ENTDAA CCC transfer, as target .......... 2330
  52.9.7 I3C broadcast DEFTGTS CCC transfer, as target .......... 2331
  52.9.8 I3C broadcast DEFGRPA CCC transfer, as target .......... 2332
  52.9.9 I3C direct GETSTATUS CCC response, as target .......... 2332
  52.9.10 I3C private read/write transfer, as controller .......... 2334
  52.9.11 I3C private read/write transfer, as target .............. 2334
  52.9.12 Legacy I2C read/write transfer, as controller .......... 2336
  52.9.13 I3C IBI transfer, as controller/target .................. 2337
  52.9.14 I3C hot-join request transfer, as controller/target ....... 2338
52.9.15 I3C controller-role request transfer, as controller/target .............. 2339
52.10 I3C FIFOs management, as controller ........................................ 2340
  52.10.1 C-FIFO management, as controller ......................................... 2340
  52.10.2 TX-FIFO management, as controller ....................................... 2341
  52.10.3 RX-FIFO management, as controller ....................................... 2344
  52.10.4 S-FIFO management, as controller ....................................... 2346
52.11 I3C FIFOs management, as target ............................................. 2348
  52.11.1 RX-FIFO management, as target ........................................... 2348
  52.11.2 TX-FIFO management, as target ........................................... 2349
52.12 I3C error management ............................................................... 2352
  52.12.1 Controller error management ................................................. 2352
  52.12.2 Target error management ..................................................... 2354
52.13 I3C wake-up from low-power mode(s) ......................................... 2355
  52.13.1 Wake-up from Stop ............................................................. 2355
52.14 I3C in low-power modes ............................................................ 2358
52.15 I3C interrupts ............................................................................. 2359
52.16 I3C registers .............................................................................. 2360
  52.16.1 I3C message control register (I3C_CR) ................................... 2360
  52.16.2 I3C message control register [alternate] (I3C_CR) .................. 2362
  52.16.3 I3C configuration register (I3C_CFG) ...................................... 2364
  52.16.4 I3C receive data byte register (I3C_RDR) .............................. 2369
  52.16.5 I3C receive data word register (I3C_RDWR) .......................... 2369
  52.16.6 I3C transmit data byte register (I3C_TDR) ............................ 2370
  52.16.7 I3C transmit data word register (I3C_TDWR) .......................... 2371
  52.16.8 I3C IBI payload data register (I3C_IBIDR) ............................ 2373
  52.16.9 I3C target transmit configuration register (I3C_TGTDR) .......... 2374
  52.16.10 I3C status register (I3C_SR) ............................................... 2375
  52.16.11 I3C status error register (I3C_SER) ....................................... 2376
  52.16.12 I3C received message register (I3C_RMR) ............................ 2378
  52.16.13 I3C event register (I3C_EVR) ............................................... 2379
  52.16.14 I3C interrupt enable register (I3C_IER) ............................... 2383
  52.16.15 I3C clear event register (I3C_CEVR) .................................... 2385
  52.16.16 I3C own device characteristics register (I3C_DEVR0) ............ 2387
  52.16.17 I3C device x characteristics register (I3C_DEVRx) ............... 2389
  52.16.18 I3C maximum read length register (I3C_MAXRLR) ............... 2391
  52.16.19 I3C maximum write length register (I3C_MAXWLR) ............. 2392
52.16.20 I3C timing register 0 (I3C_TIMINGR0) ........................................... 2393
52.16.21 I3C timing register 1 (I3C_TIMINGR1) ........................................... 2394
52.16.22 I3C timing register 2 (I3C_TIMINGR2) ........................................... 2396
52.16.23 I3C bus characteristics register (I3C_BCR) ................................. 2397
52.16.24 I3C device characteristics register (I3C_DCR) ............................. 2398
52.16.25 I3C get capability register (I3C_GETCAPR) .................................. 2399
52.16.26 I3C controller-role capability register (I3C_CRCAPR) ..................... 2400
52.16.27 I3C get max data speed register (I3C_GETMXDSR) ....................... 2401
52.16.28 I3C extended provisioned ID register (I3C_EPIDR) ......................... 2403
52.16.29 I3C register map ........................................................................ 2403

53 Universal synchronous/asynchronous receiver transmitter (USART/UART) ........................................... 2407
53.1 Introduction ................................................................................. 2407
53.2 USART main features ................................................................. 2407
53.3 USART extended features ............................................................ 2408
53.4 USART implementation ............................................................... 2408
53.5 USART functional description ....................................................... 2410
  53.5.1 USART block diagram ............................................................. 2410
  53.5.2 USART pins and internal signals .............................................. 2410
  53.5.3 USART clocks ..................................................................... 2412
  53.5.4 USART character description ................................................. 2412
  53.5.5 USART FIFOs and thresholds ............................................... 2415
  53.5.6 USART transmitter ............................................................... 2415
  53.5.7 USART receiver .................................................................... 2418
  53.5.8 USART baud rate generation .................................................. 2425
  53.5.9 Tolerance of the USART receiver to clock deviation ................. 2427
  53.5.10 USART auto baud rate detection ......................................... 2428
  53.5.11 USART multiprocessor communication .................................. 2430
  53.5.12 USART Modbus communication .......................................... 2432
  53.5.13 USART parity control .......................................................... 2433
  53.5.14 USART LIN (local interconnection network) mode ................. 2434
  53.5.15 USART synchronous mode .................................................. 2436
  53.5.16 USART single-wire half-duplex communication ...................... 2440
  53.5.17 USART receiver timeout ....................................................... 2440
  53.5.18 USART smartcard mode ....................................................... 2441
  53.5.19 USART IrDA SIR ENDEC block ............................................ 2445
53.5.20 Continuous communication using USART and DMA .......................... 2448
53.5.21 RS232 hardware flow control and RS485 driver enable ................. 2450
53.5.22 USART low-power management ................................................. 2453
53.6 USART in low-power modes .......................................................... 2456
53.7 USART interrupts ................................................................. 2456
53.8 USART registers ................................................................. 2459
  53.8.1 USART control register 1 (USART_CR1) .................................... 2459
  53.8.2 USART control register 1 [alternate] (USART_CR1) ...................... 2463
  53.8.3 USART control register 2 (USART_CR2) .................................... 2466
  53.8.4 USART control register 3 (USART_CR3) .................................... 2470
  53.8.5 USART control register 3 [alternate] (USART_CR3) ...................... 2475
  53.8.6 USART baud rate register (USART_BRR) ................................ 2478
  53.8.7 USART guard time and prescaler register (USART_GTPR) ............... 2479
  53.8.8 USART receiver timeout register (USART_RTOR) ......................... 2480
  53.8.9 USART request register (USART_RQR) ..................................... 2481
  53.8.10 USART interrupt and status register (USART_ISR) ..................... 2482
  53.8.11 USART interrupt and status register [alternate] (USART_ISR) ....... 2488
  53.8.12 USART interrupt flag clear register (USART_ICR) ..................... 2493
  53.8.13 USART receive data register (USART_RDR) .............................. 2495
  53.8.14 USART transmit data register (USART_TDR) ............................. 2495
  53.8.15 USART prescaler register (USART_PRESC) ................................ 2496
  53.8.16 USART register map .......................................................... 2497

54 Low-power universal asynchronous receiver transmitter (LPUART) ........  2499
  54.1 Introduction ................................................................. 2499
  54.2 LPUART main features ........................................................ 2499
  54.3 LPUART implementation ...................................................... 2500
  54.4 LPUART functional description .............................................. 2502
    54.4.1 LPUART block diagram .................................................. 2502
    54.4.2 LPUART pins and internal signals .................................... 2503
    54.4.3 LPUART clocks .......................................................... 2504
    54.4.4 LPUART character description ........................................ 2504
    54.4.5 LPUART FIFOs and thresholds ....................................... 2506
    54.4.6 LPUART transmitter .................................................... 2506
    54.4.7 LPUART receiver ....................................................... 2510
54.4.8 LPUART baud rate generation ........................................ 2514
54.4.9 Tolerance of the LPUART receiver to clock deviation .......... 2516
54.4.10 LPUART multiprocessor communication .......................... 2517
54.4.11 LPUART parity control ............................................ 2519
54.4.12 LPUART single-wire half-duplex communication ............... 2520
54.4.13 Continuous communication using DMA and LPUART .......... 2520
54.4.14 RS232 hardware flow control and RS485 driver enable ....... 2523
54.4.15 LPUART low-power management .................................. 2525
54.5 LPUART in low-power modes ......................................... 2528
54.6 LPUART interrupts ..................................................... 2529
54.7 LPUART registers ....................................................... 2530
  54.7.1 LPUART control register 1 (LPUART_CR1) ..................... 2530
  54.7.2 LPUART control register 1 [alternate] (LPUART_CR1) ........ 2533
  54.7.3 LPUART control register 2 (LPUART_CR2) .................... 2536
  54.7.4 LPUART control register 3 (LPUART_CR3) .................... 2538
  54.7.5 LPUART control register 3 [alternate] (LPUART_CR3) ....... 2541
  54.7.6 LPUART baud rate register (LPUART_BRR) ................. 2544
  54.7.7 LPUART request register (LPUART_RQR) ...................... 2544
  54.7.8 LPUART interrupt and status register (LPUART_ISR) ........ 2545
  54.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) .... 2549
  54.7.10 LPUART interrupt flag clear register (LPUART_ICR) ....... 2552
  54.7.11 LPUART receive data register (LPUART_RDR) ............. 2553
  54.7.12 LPUART transmit data register (LPUART_TDR) ............ 2554
  54.7.13 LPUART prescaler register (LPUART_PRESC) .............. 2554
  54.7.14 LPUART register map ........................................... 2555

55 Serial peripheral interface (SPI) ..................................... 2557
  55.1 Introduction ......................................................... 2557
  55.2 SPI main features .................................................. 2557
  55.3 SPI implementation ................................................ 2558
  55.4 SPI functional description ....................................... 2559
    55.4.1 SPI block diagram .......................................... 2559
    55.4.2 SPI pins and internal signals ............................... 2560
    55.4.3 SPI communication general aspects ......................... 2561
    55.4.4 Communications between one master and one slave .......... 2561
    55.4.5 Standard multislave communication .......................... 2564
55.11.1 SPI/I2S control register 1 (SPI_CR1) ........................................... 2613
55.11.2 SPI/I2S control register 2 (SPI_CR2) ........................................ 2615
55.11.3 SPI/I2S configuration register 1 (SPI_CFG1) ................................ 2616
55.11.4 SPI/I2S configuration register 2 (SPI_CFG2) ................................ 2619
55.11.5 SPI/I2S interrupt enable register (SPI_IER) .................................. 2622
55.11.6 SPI/I2S status register (SPI_SR) .................................................. 2623
55.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) ...................... 2625
55.11.8 SPI/I2S transmit data register (SPI_TXDR) ...................................... 2626
55.11.9 SPI/I2S receive data register (SPI_RXDR) ....................................... 2627
55.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) .................................. 2627
55.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) ............................... 2628
55.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) ................................... 2629
55.11.13 SPI/I2S underrun data register (SPI_UDRDR) .................................. 2629
55.11.14 SPI/I2S configuration register (SPI_I2SCFGR) ................................ 2630
55.11.15 SPI/I2S register map ............................................................... 2632

56 Serial audio interface (SAI) .............................................................. 2634
56.1 Introduction ..................................................................................... 2634
56.2 SAI main features .......................................................................... 2634
56.3 SAI implementation ........................................................................ 2635
56.4 SAI functional description .............................................................. 2635
  56.4.1 SAI block diagram ..................................................................... 2635
  56.4.2 SAI pins and internal signals ..................................................... 2637
  56.4.3 Main SAI modes ....................................................................... 2637
  56.4.4 SAI synchronization mode ......................................................... 2638
  56.4.5 Audio data size ....................................................................... 2639
  56.4.6 Frame synchronization ............................................................. 2640
  56.4.7 Slot configuration ..................................................................... 2643
  56.4.8 SAI clock generator ................................................................. 2645
  56.4.9 Internal FIFOs ......................................................................... 2648
  56.4.10 PDM interface ....................................................................... 2650
  56.4.11 AC’97 link controller ............................................................... 2658
  56.4.12 SPDIF output ........................................................................... 2660
  56.4.13 Specific features ..................................................................... 2662
  56.4.14 Error flags ............................................................................. 2666
  56.4.15 Disabling the SAI .................................................................. 2669
  56.4.16 SAI DMA interface ................................................................. 2669
56.5 SAI interrupts ............................................................. 2670
56.6 SAI registers ............................................................ 2672
  56.6.1 SAI global configuration register (SAI_GCR) ............... 2672
  56.6.2 SAI configuration register 1 (SAI_ACR1) ..................... 2672
  56.6.3 SAI configuration register 2 (SAI_ACR2) ..................... 2675
  56.6.4 SAI frame configuration register (SAI_AFRCR) ............. 2677
  56.6.5 SAI slot register (SAI_A_SLOTR) ......................... 2678
  56.6.6 SAI interrupt mask register (SAI_AIM) ...................... 2679
  56.6.7 SAI status register (SAI_ASR) ............................... 2680
  56.6.8 SAI clear flag register (SAI_A_CLRFR) ..................... 2682
  56.6.9 SAI data register (SAI_ADR) .................................. 2683
  56.6.10 SAI configuration register 1 (SAI_BCR1) .................. 2684
  56.6.11 SAI configuration register 2 (SAI_BCR2) .................. 2686
  56.6.12 SAI frame configuration register (SAI_BFRCR) .......... 2688
  56.6.13 SAI slot register (SAI_B_SLOTR) ........................... 2689
  56.6.14 SAI interrupt mask register (SAI_BIM) .................... 2690
  56.6.15 SAI status register (SAI_BSR) .............................. 2692
  56.6.16 SAI clear flag register (SAI_B_CLRFR) ................... 2694
  56.6.17 SAI data register (SAI_BDR) ............................... 2695
  56.6.18 SAI PDM control register (SAI_PDMCR) ................. 2695
  56.6.19 SAI PDM delay register (SAI_PDM_DLY) .................. 2696
  56.6.20 SAI register map ............................................. 2698

57 SPDIF receiver interface (SPDIFRX) .............................. 2700
  57.1 SPDIFRX interface introduction ............................... 2700
  57.2 SPDIFRX main features ......................................... 2700
  57.3 SPDIFRX functional description ............................... 2700
    57.3.1 SPDIFRX pins and internal signals ...................... 2701
    57.3.2 S/PDIF protocol (IEC-60958) .......................... 2702
    57.3.3 SPDIFRX decoder (SPDIFRX_DC) ....................... 2704
    57.3.4 SPDIFRX tolerance to clock deviation .................. 2708
    57.3.5 SPDIFRX synchronization ............................... 2708
    57.3.6 SPDIFRX handling ....................................... 2710
    57.3.7 Data reception management .............................. 2712
    57.3.8 Dedicated control flow .................................. 2714
    57.3.9 Reception errors ........................................ 2715
    57.3.10 Clocking strategy ..................................... 2717
57.3.11 Symbol clock generation ........................................ 2717
57.3.12 DMA interface .................................................. 2719
57.3.13 Interrupt generation ........................................... 2720
57.3.14 Register protection ............................................. 2721
57.4 Programming procedures ......................................... 2721
  57.4.1 Initialization phase ........................................... 2722
  57.4.2 Handling of interrupts coming from SPDIFRX ............... 2723
  57.4.3 Handling of interrupts coming from DMA .................... 2723
57.5 SPDIFRX interface registers ..................................... 2724
  57.5.1 SPDIFRX control register (SPDIFRX_CR) .................... 2724
  57.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR) .......... 2726
  57.5.3 SPDIFRX status register (SPDIFRX_SR) ..................... 2727
  57.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR) .. 2729
  57.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR) ......... 2730
  57.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR) ......... 2730
  57.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR) ......... 2731
  57.5.8 SPDIFRX channel status register (SPDIFRX_CSR) ......... 2732
  57.5.9 SPDIFRX debug information register (SPDIFRX_DIR) ...... 2732
  57.5.10 SPDIFRX interface register map ........................... 2733

58 Management data input/output (MDIOS) .......................... 2734
  58.1 MDIOS introduction ............................................. 2734
  58.2 MDIOS main features .......................................... 2734
  58.3 MDIOS functional description ................................... 2735
    58.3.1 MDIOS block diagram ...................................... 2735
    58.3.2 MDIOS pins and internal signals .......................... 2735
    58.3.3 MDIOS protocol ........................................... 2735
    58.3.4 MDIOS enabling and disabling ............................. 2736
    58.3.5 MDIOS data ................................................ 2737
    58.3.6 MDIOS APB frequency ...................................... 2738
    58.3.7 Write/read flags and interrupts ........................... 2738
    58.3.8 MDIOS error management ................................... 2739
    58.3.9 MDIOS in Stop mode ....................................... 2740
    58.3.10 MDIOS interrupts ......................................... 2740
  58.4 MDIOS registers ................................................ 2740
    58.4.1 MDIOS configuration register (MDIOS_CR) ............... 2740
58.4.2 MDIOS write flag register (MDIOS_WRFR) ........................................... 2741
58.4.3 MDIOS clear write flag register (MDIOS_CWRFR) .............................. 2742
58.4.4 MDIOS read flag register (MDIOS_RDFR) ........................................... 2742
58.4.5 MDIOS clear read flag register (MDIOS_CRDFR) ............................... 2743
58.4.6 MDIOS status register (MDIOS_SR) ...................................................... 2743
58.4.7 MDIOS clear flag register (MDIOS_CLRFR) .......................................... 2744
58.4.8 MDIOS input data register x (MDIOS_DINRx) ...................................... 2744
58.4.9 MDIOS output data register x (MDIOS_DOUTRx) .................................. 2745
58.4.10 MDIOS register map ........................................................................ 2745

59 Secure digital input/output MultiMediaCard interface (SDMMC) ............. 2747

59.1 SDMMC main features ........................................................................... 2747
59.2 SDMMC implementation ....................................................................... 2747
59.3 SDMMC bus topology .......................................................................... 2748
59.4 SDMMC operation modes ..................................................................... 2750
59.5 SDMMC functional description ............................................................. 2751
  59.5.1 SDMMC block diagram .................................................................. 2751
  59.5.2 SDMMC pins and internal signals .................................................... 2751
  59.5.3 General description ........................................................................ 2752
  59.5.4 SDMMC adapter ............................................................................ 2754
  59.5.5 SDMMC AHB slave interface ......................................................... 2776
  59.5.6 SDMMC AHB master interface ....................................................... 2777
  59.5.7 AHB and SDMMC_CK clock relation ............................................. 2780
59.6 Card functional description .................................................................. 2780
  59.6.1 SD I/O mode .................................................................................. 2780
  59.6.2 CMD12 send timing ....................................................................... 2788
  59.6.3 Sleep (CMD5) ................................................................................ 2792
  59.6.4 Interrupt mode (Wait-IRQ) .............................................................. 2793
  59.6.5 Boot operation ................................................................................ 2794
  59.6.6 Response R1b handling ................................................................. 2797
  59.6.7 Reset and card cycle power ............................................................. 2798
59.7 Hardware flow control .......................................................................... 2799
59.8 Ultra-high-speed phase I (UHS-I) voltage switch .................................. 2799
59.9 SDMMC interrupts .............................................................................. 2802
59.10 SDMMC registers ................................................................................ 2804
  59.10.1 SDMMC power control register (SDMMC_POWER) ....................... 2804
## 59.10.2 SDMMC clock control register (SDMMC_CLKCR)

59.10.3 SDMMC argument register (SDMMC_ARGR)

59.10.4 SDMMC command register (SDMMC_CMDR)

59.10.5 SDMMC command response register (SDMMC_RESPCMDR)

59.10.6 SDMMC response x register (SDMMC_RESPxR)

59.10.7 SDMMC data timer register (SDMMC_DTIMER)

59.10.8 SDMMC data length register (SDMMC_DLENR)

59.10.9 SDMMC data control register (SDMMC_DCTRL)

59.10.10 SDMMC data counter register (SDMMC_DCCTR)

59.10.11 SDMMC status register (SDMMC_STAR)

59.10.12 SDMMC interrupt clear register (SDMMC_ICR)

59.10.13 SDMMC mask register (SDMMC_MASKR)

59.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

59.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR)

59.10.16 SDMMC DMA buffer size register (SDMMC_IDMABSIZER)

59.10.17 SDMMC DMA buffer base address register (SDMMC_IDMABASER)

59.10.18 SDMMC DMA linked list address register (SDMMC_IDMALAR)

59.10.19 SDMMC DMA linked list memory base register (SDMMC_IDMABAR)

59.10.20 SDMMC data FIFO registers x (SDMMC_FIFORx)

59.10.21 SDMMC register map

## 60 FD controller area network (FDCAN)

### 60.1 Introduction

### 60.2 FDCAN main features

### 60.3 FDCAN functional description

60.3.1 Bit timing

60.3.2 Operating modes

60.3.3 Message RAM

60.3.4 FIFO acknowledge handling

60.3.5 FDCAN Rx FIFO element

60.3.6 FDCAN Tx buffer element

60.3.7 FDCAN Tx event FIFO element

60.3.8 FDCAN standard message ID filter element

60.3.9 FDCAN extended message ID filter element

### 60.4 FDCAN registers
60.4.1  FDCAN core release register (FDCAN_CREL) ........................................ 2861
60.4.2  FDCAN endian register (FDCAN_ENDN) .................................................... 2861
60.4.3  FDCAN data bit timing and prescaler register (FDCAN_DBTP) ................... 2861
60.4.4  FDCAN test register (FDCAN_TEST) ....................................................... 2862
60.4.5  FDCAN RAM watchdog register (FDCAN_RWD) ......................................... 2863
60.4.6  FDCAN CC control register (FDCAN_CCCR) ............................................. 2864
60.4.7  FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) ............. 2865
60.4.8  FDCAN timestamp counter configuration register (FDCAN_TSCC) ............. 2867
60.4.9  FDCAN timestamp counter value register (FDCAN_TSCV) ......................... 2867
60.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC) ............. 2868
60.4.11 FDCAN timeout counter value register (FDCAN_TOCV) ............................ 2869
60.4.12 FDCAN error counter register (FDCAN_ECR) ........................................... 2869
60.4.13 FDCAN protocol status register (FDCAN_PSR) ....................................... 2870
60.4.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) .......... 2872
60.4.15 FDCAN interrupt register (FDCAN_IR) .................................................... 2872
60.4.16 FDCAN interrupt enable register (FDCAN_IE) ......................................... 2875
60.4.17 FDCAN interrupt line select register (FDCAN_ILS) ................................ 2877
60.4.18 FDCAN interrupt line enable register (FDCAN_ILE) ................................. 2878
60.4.19 FDCAN global filter configuration register (FDCAN_RXGFC) ................. 2878
60.4.20 FDCAN extended ID and mask register (FDCAN_XIDAM) .......................... 2880
60.4.21 FDCAN high-priority message status register (FDCAN_HPMS) ................ 2880
60.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) .................................. 2881
60.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) .......................... 2882
60.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) .................................. 2882
60.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) ........................ 2883
60.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC) .......................... 2883
60.4.27 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) ........................... 2884
60.4.28 FDCAN Tx buffer request pending register (FDCAN_TXBRP) ...................... 2884
60.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR) ........................... 2885
60.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) ............ 2886
60.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) ........... 2886
60.4.32 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) ............ 2887
60.4.33 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) .................. 2887
60.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) .................. 2888
60.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS) ............................ 2888
60.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) ................. 2889
61 USB on-the-go full-speed (OTG_FS) .............................. 2894
61.1 Introduction .................................................. 2894
61.2 OTG_FS main features ........................................ 2895
   61.2.1 General features ....................................... 2895
   61.2.2 Host-mode features .................................... 2896
   61.2.3 Peripheral-mode features ............................... 2896
61.3 OTG_FS implementation ....................................... 2897
61.4 OTG_FS functional description ............................... 2898
   61.4.1 OTG_FS block diagram .................................. 2898
   61.4.2 OTG_FS pin and internal signals ....................... 2898
   61.4.3 OTG_FS core ............................................. 2899
   61.4.4 Embedded full-speed OTG PHY connected to OTG_FS .... 2899
   61.4.5 OTG detections ........................................... 2900
61.5 OTG_FS dual role device (DRD) .............................. 2900
   61.5.1 ID line detection ....................................... 2900
61.6 OTG_FS as a USB peripheral ................................ 2901
   61.6.1 Peripheral states ....................................... 2901
   61.6.2 Peripheral endpoints ................................... 2902
61.7 OTG_FS as a USB host ........................................ 2905
   61.7.1 USB host states ......................................... 2905
   61.7.2 Host channels .......................................... 2906
   61.7.3 Host scheduler ......................................... 2908
61.8 OTG_FS SOF trigger .......................................... 2909
   61.8.1 Host SOFs .............................................. 2909
   61.8.2 Peripheral SOFs ........................................ 2909
61.9 OTG_FS low-power modes .................................... 2910
61.10 OTG_FS Dynamic update of the OTG_HFIR register ........ 2911
61.11 OTG_FS data FIFOs .......................................... 2911
   61.11.1 Peripheral FIFO architecture ......................... 2912
   61.11.2 Host FIFO architecture ............................... 2913
   61.11.3 FIFO RAM allocation ................................... 2914
61.12 OTG_FS system performance ................................ 2916
61.13 OTG_FS interrupts .......................................... 2916
61.14  OTG_FS control and status registers .............................. 2918
  61.14.1 CSR memory map ........................................... 2918
61.15  OTG_FS registers .................................................. 2923
  61.15.1 OTG control and status register (OTG_GOTGCTL) .......... 2923
  61.15.2 OTG interrupt register (OTG_GOTGINT) .................... 2925
  61.15.3 OTG AHB configuration register (OTG_GAHBCFG) ......... 2926
  61.15.4 OTG USB configuration register (OTG_GUSBCFG) .......... 2927
  61.15.5 OTG reset register (OTG_GRSTCTL) ......................... 2928
  61.15.6 OTG core interrupt register [alternate] (OTG_GINTSTS) ... 2931
  61.15.7 OTG core interrupt register [alternate] (OTG_GINTSTS) ... 2935
  61.15.8 OTG interrupt mask register [alternate] (OTG_GINTMSK) ... 2939
  61.15.9 OTG interrupt mask register [alternate] (OTG_GINTMSK) ... 2940
  61.15.10 OTG receive status debug read [alternate] (OTG_GRXSTSR) .. 2942
  61.15.11 OTG receive status debug read register [alternate]
             (OTG_GRXSTSR) ........................................ 2943
  61.15.12 OTG status read and pop registers [alternate] (OTG_GRXSTSP) 2944
  61.15.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP) 2945
  61.15.14 OTG receive FIFO size register (OTG_GRXFSIZ) .......... 2946
  61.15.15 OTG host non-periodic transmit FIFO size register [alternate]
             (OTG_HNPTXFSIZ) ....................................... 2947
  61.15.16 Endpoint 0 Transmit FIFO size [alternate] (OTG_DIEPTXF0) ... 2947
  61.15.17 OTG non-periodic transmit FIFO/queue status register
             (OTG_HNPTXSTS) ........................................ 2948
  61.15.18 OTG general core configuration register (OTG_GCCFG) .... 2949
  61.15.19 OTG core ID register (OTG_CID) ......................... 2950
  61.15.20 OTG core LPM configuration register (OTG_GLPMCFG) .... 2950
  61.15.21 OTG host periodic transmit FIFO size register
             (OTG_HPTXFSIZ) ........................................ 2955
  61.15.22 OTG device IN endpoint transmit FIFO x size register
             (OTG_DIEPTXFx) ....................................... 2955
  61.15.23 Host-mode registers ...................................... 2955
  61.15.24 OTG host configuration register (OTG_HCFG) ............ 2956
  61.15.25 OTG host frame interval register (OTG_HFIR) .......... 2956
  61.15.26 OTG host frame number/frame time remaining register
             (OTG_HFNUM) ........................................ 2957
  61.15.27 OTG_Host periodic transmit FIFO/queue status register
             (OTG_HPTXSTS) ....................................... 2958
  61.15.28 OTG host all channels interrupt register (OTG_HAINT) ... 2959
61.15.29 OTG host all channels interrupt mask register (OTG_HAINTMSK) ........................................ 2959
61.15.30 OTG host port control and status register (OTG_HPRT) ......................................................... 2960
61.15.31 OTG host channel x characteristics register (OTG_HCCCHARx) ........................................... 2962
61.15.32 OTG host channel x interrupt register (OTG_HCINTx) ............................................................ 2963
61.15.33 OTG host channel x interrupt mask register (OTG_HCINTMSKx) ............................................ 2964
61.15.34 OTG host channel x transfer size register (OTG_HCTSIZx) ..................................................... 2965
61.15.35 Device-mode registers ........................................................................................................... 2966
61.15.36 OTG device configuration register (OTG_DCFG) ................................................................. 2966
61.15.37 OTG device control register (OTG_DCTL) .............................................................................. 2967
61.15.38 OTG device status register (OTG_DSTS) .................................................................................. 2970
61.15.39 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK) ............................ 2971
61.15.40 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK) ...................... 2972
61.15.41 OTG device all endpoints interrupt register (OTG_DAINT) ...................................................... 2973
61.15.42 OTG all endpoints interrupt mask register (OTG_DAINTMSK) .................................................. 2974
61.15.43 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) .............. 2974
61.15.44 OTG device each endpoint interrupt register (OTG_DEACHINT) ............................................. 2975
61.15.45 OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK) ............................ 2975
61.15.46 OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1) .............. 2976
61.15.47 OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1) .......... 2977
61.15.48 OTG device control IN endpoint 0 control register (OTG_DIEPCTL0) ...................................... 2978
61.15.49 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) ................................. 2980
61.15.50 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) ................................. 2982
61.15.51 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) .............................................. 2984
61.15.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) ....................................... 2985
61.15.53 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) .............................. 2986
61.15.54 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) ........................................ 2987
61.15.55 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) ............................... 2988
61.15.56 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) . . 2989
61.15.57 OTG device OUT endpoint 0 transfer size register
(OTG_DOEPTSIZ0) ........................................................... 2990
61.15.58 OTG device OUT endpoint x control register [alternate]
(OTG_DOEPCTLx) ........................................................... 2991
61.15.59 OTG device OUT endpoint x control register [alternate]
(OTG_DOEPCTLx) ........................................................... 2993
61.15.60 OTG device OUT endpoint x transfer size register
(OTG_DOEPTSIZx) ........................................................... 2995
61.15.61 OTG power and clock gating control register (OTG_PCGCCTL) . . 2996
61.15.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1) . 2997
61.15.63 OTG_FS register map .................................................. 2998

61.16 OTG_FS programming model ............................................ 3006
61.16.1 Core initialization ....................................................... 3006
61.16.2 Host initialization ....................................................... 3007
61.16.3 Device initialization ..................................................... 3007
61.16.4 Host programming model .............................................. 3008
61.16.5 Device programming model .......................................... 3029
61.16.6 Worst case response time ............................................. 3050
61.16.7 OTG programming model .............................................. 3052

62 USB on-the-go high-speed (OTG_HS) ...................................... 3053
62.1 Introduction ................................................................. 3053
62.2 OTG_HS main features .................................................... 3054
  62.2.1 General features ....................................................... 3054
  62.2.2 Host-mode features ................................................... 3055
  62.2.3 Peripheral-mode features ........................................... 3055
62.3 OTG_HS implementation .................................................. 3055
62.4 OTG_HS functional description ......................................... 3056
  62.4.1 OTG_HS block diagram .............................................. 3056
  62.4.2 OTG_HS pin and internal signals ................................... 3056
  62.4.3 OTG_HS core .......................................................... 3057
  62.4.4 OTG detections ........................................................ 3057
  62.4.5 High-speed OTG PHY connected to OTG_HS ...................... 3057
62.5 OTG_HS dual role device (DRD) ........................................ 3058
  62.5.1 ID line detection ...................................................... 3058
62.6 OTG_HS as a USB peripheral ............................................ 3058
62.6.1 Peripheral states ............................................. 3059
62.6.2 Peripheral endpoints ........................................... 3060
62.7 OTG_HS as a USB host ........................................... 3062
  62.7.1 USB host states ............................................. 3063
  62.7.2 Host channels .............................................. 3064
  62.7.3 Host scheduler .............................................. 3065
62.8 OTG_HS SOF trigger ............................................. 3066
  62.8.1 Host SOFs .................................................. 3067
  62.8.2 Peripheral SOFs ............................................. 3067
62.9 OTG_HS low-power modes ....................................... 3067
62.10 OTG_HS Dynamic update of the OTG_HFIR register .......... 3068
62.11 OTG_HS data FIFOs ............................................ 3069
  62.11.1 Peripheral FIFO architecture ............................. 3070
  62.11.2 Host FIFO architecture .................................... 3071
  62.11.3 FIFO RAM allocation ....................................... 3072
62.12 OTG_HS interrupts ............................................. 3074
62.13 OTG_HS control and status registers ......................... 3076
  62.13.1 CSR memory map .......................................... 3076
62.14 OTG_HS registers ............................................. 3081
  62.14.1 OTG control and status register (OTG_GOTGCTL) .......... 3081
  62.14.2 OTG interrupt register (OTG_GOTGINT) .................... 3083
  62.14.3 OTG AHB configuration register (OTG_GAHCFCFG) ........ 3084
  62.14.4 OTG USB configuration register (OTG_GUSBCCFG) ........ 3085
  62.14.5 OTG reset register (OTG_GRSTCTL) ........................ 3087
  62.14.6 OTG core interrupt register [alternate] (OTG_GINTSTS) .. 3090
  62.14.7 OTG core interrupt register [alternate] (OTG_GINTSTS) .. 3094
  62.14.8 OTG interrupt mask register [alternate] (OTG_GINTMSK) .. 3099
  62.14.9 OTG interrupt mask register [alternate] (OTG_GINTMSK) .. 3100
  62.14.10 OTG receive status debug read register [alternate]
             (OTG_GRXSTSR) ........................................... 3102
  62.14.11 OTG receive status debug read register [alternate]
             (OTG_GRXSTSR) ........................................... 3103
  62.14.12 OTG status read and pop registers (OTG_GRXSTSP) ....... 3104
  62.14.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP) .... 3105
  62.14.14 OTG receive FIFO size register (OTG_GRXFSIZ) ........ 3106
  62.14.15 OTG host non-periodic transmit FIFO size register [alternate]
             (OTG_HNPTXFSIZ) ....................................... 3107
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>62.14.16</td>
<td>Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)</td>
</tr>
<tr>
<td>62.14.17</td>
<td>OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)</td>
</tr>
<tr>
<td>62.14.18</td>
<td>OTG general core configuration register (OTG_GCCFG)</td>
</tr>
<tr>
<td>62.14.19</td>
<td>OTG core ID register (OTG_CID)</td>
</tr>
<tr>
<td>62.14.20</td>
<td>OTG core LPM configuration register (OTG_GLPMCFG)</td>
</tr>
<tr>
<td>62.14.21</td>
<td>OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)</td>
</tr>
<tr>
<td>62.14.22</td>
<td>OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)</td>
</tr>
<tr>
<td>62.14.23</td>
<td>Host-mode registers</td>
</tr>
<tr>
<td>62.14.24</td>
<td>OTG host configuration register (OTG_HCFG)</td>
</tr>
<tr>
<td>62.14.25</td>
<td>OTG host frame interval register (OTG_HFIR)</td>
</tr>
<tr>
<td>62.14.26</td>
<td>OTG host frame number/frame time remaining register (OTG_HFNUM)</td>
</tr>
<tr>
<td>62.14.27</td>
<td>OTG Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)</td>
</tr>
<tr>
<td>62.14.28</td>
<td>OTG host all channels interrupt register (OTG_HAINT)</td>
</tr>
<tr>
<td>62.14.29</td>
<td>OTG host all channels interrupt mask register (OTG_HAINTMSK)</td>
</tr>
<tr>
<td>62.14.30</td>
<td>OTG host port control and status register (OTG_HPRT)</td>
</tr>
<tr>
<td>62.14.31</td>
<td>OTG host channel x characteristics register (OTG_HCCHARx)</td>
</tr>
<tr>
<td>62.14.32</td>
<td>OTG host channel x split control register (OTG_HCSPLTx)</td>
</tr>
<tr>
<td>62.14.33</td>
<td>OTG host channel x interrupt register (OTG_HCINTx)</td>
</tr>
<tr>
<td>62.14.34</td>
<td>OTG host channel x interrupt mask register (OTG_HCINTMSKx)</td>
</tr>
<tr>
<td>62.14.35</td>
<td>OTG host channel x transfer size register (OTG_HCTSIZx)</td>
</tr>
<tr>
<td>62.14.36</td>
<td>OTG host channel x DMA address register (OTG_HCDMAx)</td>
</tr>
<tr>
<td>62.14.37</td>
<td>Device-mode registers</td>
</tr>
<tr>
<td>62.14.38</td>
<td>OTG device configuration register (OTG_DCFG)</td>
</tr>
<tr>
<td>62.14.39</td>
<td>OTG device control register (OTG_DCTL)</td>
</tr>
<tr>
<td>62.14.40</td>
<td>OTG device status register (OTG_DSTS)</td>
</tr>
<tr>
<td>62.14.41</td>
<td>OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</td>
</tr>
<tr>
<td>62.14.42</td>
<td>OTG device OUT endpoint common interrupt mask register (OTG_DOEPMASK)</td>
</tr>
<tr>
<td>62.14.43</td>
<td>OTG device all endpoints interrupt register (OTG_DAINFT)</td>
</tr>
<tr>
<td>62.14.44</td>
<td>OTG all endpoints interrupt mask register (OTG_DAINFTMSK)</td>
</tr>
<tr>
<td>62.14.45</td>
<td>OTG device threshold control register (OTG_DTHRTC)</td>
</tr>
</tbody>
</table>
62.14.46 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) ........................................... 3136
62.14.47 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) ......................................................... 3137
62.14.48 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) ......................................................... 3139
62.14.49 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) ................................................................. 3141
62.14.50 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) ............................................................. 3142
62.14.51 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx) ............................................................ 3143
62.14.52 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) ......................................................... 3143
62.14.53 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) ............................................................ 3144
62.14.54 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) ......................................................... 3144
62.14.55 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) ............................................................ 3146
62.14.56 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) ............................................................. 3148
62.14.57 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx) ............................................................ 3149
62.14.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) ......................................................... 3149
62.14.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) ......................................................... 3151
62.14.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) ............................................................ 3153
62.14.61 OTG power and clock gating control register (OTG_PCGCCTL) ................................................................. 3154
62.14.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1) ............................................................. 3155
62.14.63 OTG_HS register map .......................................................................................................................... 3156

62.15 OTG_HS programming model .................................................................................................................. 3164
62.15.1 Core initialization .................................................................................................................................. 3164
62.15.2 Host initialization .................................................................................................................................. 3165
62.15.3 Device initialization ............................................................................................................................. 3166
62.15.4 DMA mode ........................................................................................................................................... 3166
62.15.5 Host programming model .................................................................................................................. 3166
62.15.6 Device programming model .............................................................................................................. 3199
62.15.7 Worst case response time .................................................................................................................. 3219
62.15.8 OTG programming model .................................................................................................................. 3221

63 USB Type-C®/USB Power Delivery interface (UCPD) ................................................................. 3222
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>63.1</td>
<td>Introduction</td>
<td>3222</td>
</tr>
<tr>
<td>63.2</td>
<td>UCPD main features</td>
<td>3222</td>
</tr>
<tr>
<td>63.3</td>
<td>UCPD implementation</td>
<td>3222</td>
</tr>
<tr>
<td>63.4</td>
<td>UCPD functional description</td>
<td>3223</td>
</tr>
<tr>
<td>63.4.1</td>
<td>UCPD block diagram</td>
<td>3224</td>
</tr>
<tr>
<td>63.4.2</td>
<td>UCPD reset and clocks</td>
<td>3225</td>
</tr>
<tr>
<td>63.4.3</td>
<td>Physical layer protocol</td>
<td>3226</td>
</tr>
<tr>
<td>63.4.4</td>
<td>UCPD BMC transmitter</td>
<td>3232</td>
</tr>
<tr>
<td>63.4.5</td>
<td>UCPD BMC receiver</td>
<td>3234</td>
</tr>
<tr>
<td>63.4.6</td>
<td>UCPD Type-C pull-ups (Rp) and pull-downs (Rd)</td>
<td>3235</td>
</tr>
<tr>
<td>63.4.7</td>
<td>UCPD Type-C voltage monitoring and de-bouncing</td>
<td>3236</td>
</tr>
<tr>
<td>63.4.8</td>
<td>UCPD fast role swap (FRS)</td>
<td>3236</td>
</tr>
<tr>
<td>63.4.9</td>
<td>UCPD DMA Interface</td>
<td>3236</td>
</tr>
<tr>
<td>63.4.10</td>
<td>Wake-up from Stop mode</td>
<td>3236</td>
</tr>
<tr>
<td>63.5</td>
<td>UCPD programming sequences</td>
<td>3237</td>
</tr>
<tr>
<td>63.5.1</td>
<td>Initialization phase</td>
<td>3237</td>
</tr>
<tr>
<td>63.5.2</td>
<td>Type-C state machine handling</td>
<td>3237</td>
</tr>
<tr>
<td>63.5.3</td>
<td>USB PD transmit</td>
<td>3239</td>
</tr>
<tr>
<td>63.5.4</td>
<td>USB PD receive</td>
<td>3240</td>
</tr>
<tr>
<td>63.5.5</td>
<td>UCPD software trimming</td>
<td>3241</td>
</tr>
<tr>
<td>63.6</td>
<td>UCPD low-power modes</td>
<td>3241</td>
</tr>
<tr>
<td>63.7</td>
<td>UCPD interrupts</td>
<td>3242</td>
</tr>
<tr>
<td>63.8</td>
<td>UCPD registers</td>
<td>3243</td>
</tr>
<tr>
<td>63.8.1</td>
<td>UCPD configuration register 1 (UCPD_CFGR1)</td>
<td>3243</td>
</tr>
<tr>
<td>63.8.2</td>
<td>UCPD configuration register 2 (UCPD_CFGR2)</td>
<td>3245</td>
</tr>
<tr>
<td>63.8.3</td>
<td>UCPD configuration register 3 (UCPD_CFGR3)</td>
<td>3246</td>
</tr>
<tr>
<td>63.8.4</td>
<td>UCPD control register (UCPD_CR)</td>
<td>3246</td>
</tr>
<tr>
<td>63.8.5</td>
<td>UCPD interrupt mask register (UCPD_IMR)</td>
<td>3249</td>
</tr>
<tr>
<td>63.8.6</td>
<td>UCPD status register (UCPD_SR)</td>
<td>3250</td>
</tr>
<tr>
<td>63.8.7</td>
<td>UCPD interrupt clear register (UCPD_ICR)</td>
<td>3253</td>
</tr>
<tr>
<td>63.8.8</td>
<td>UCPD Tx ordered set type register (UCPD_TX_ORDSETR)</td>
<td>3254</td>
</tr>
<tr>
<td>63.8.9</td>
<td>UCPD Tx payload size register (UCPD_TX_PAYSZR)</td>
<td>3255</td>
</tr>
<tr>
<td>63.8.10</td>
<td>UCPD Tx data register (UCPD_TXDR)</td>
<td>3255</td>
</tr>
<tr>
<td>63.8.11</td>
<td>UCPD Rx ordered set register (UCPD_RX_ORDSETR)</td>
<td>3256</td>
</tr>
<tr>
<td>63.8.12</td>
<td>UCPD Rx payload size register (UCPD_RX_PAYSZR)</td>
<td>3257</td>
</tr>
<tr>
<td>63.8.13</td>
<td>UCPD receive data register (UCPD_RXDR)</td>
<td>3257</td>
</tr>
</tbody>
</table>
64  Ethernet (ETH): media access control (MAC) with DMA controller  

64.1  Ethernet introduction  ........................................ 3261
64.2  Ethernet main features  ..................................... 3261
  64.2.1  Standard compliance .................................... 3261
  64.2.2  MAC features ............................................ 3261
  64.2.3  Transaction layer (MTL) features .................... 3263
  64.2.4  DMA block features .................................... 3264
  64.2.5  Bus interface features ................................. 3264
64.3  Ethernet pins and internal signals  ......................... 3265
64.4  Ethernet architecture ...................................... 3266
  64.4.1  DMA controller .......................................... 3267
  64.4.2  MTL ...................................................... 3276
  64.4.3  MAC ...................................................... 3276
64.5  Ethernet functional description: MAC  .................... 3281
  64.5.1  Double VLAN processing ................................ 3281
  64.5.2  Source address and VLAN insertion, replacement, or deletion  . 3282
  64.5.3  Packet filtering ........................................ 3284
  64.5.4  IEEE 1588 timestamp support .......................... 3290
  64.5.5  Checksum offload engine ................................ 3315
  64.5.6  TCP segmentation offload ................................ 3321
  64.5.7  IPv4 ARP offload ....................................... 3327
  64.5.8  Loopback ................................................ 3328
  64.5.9  Flow control ............................................ 3329
  64.5.10 MAC management counters .............................. 3332
  64.5.11 Interrupts generated by the MAC ...................... 3334
  64.5.12 MAC and MMC register descriptions ................... 3334
64.6  Ethernet functional description: PHY interfaces .......... 3335
  64.6.1  Station management agent (SMA) ....................... 3335
  64.6.2  Media independent interface (MII) .................... 3342
  64.6.3  Reduced media independent interface (RMII) .......... 3343
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>64.7</td>
<td>Ethernet low-power modes</td>
</tr>
<tr>
<td>64.7.1</td>
<td>Low-power management</td>
</tr>
<tr>
<td>64.7.2</td>
<td>Energy Efficient Ethernet (EEE)</td>
</tr>
<tr>
<td>64.8</td>
<td>Ethernet interrupts</td>
</tr>
<tr>
<td>64.8.1</td>
<td>DMA interrupts</td>
</tr>
<tr>
<td>64.8.2</td>
<td>MTL interrupts</td>
</tr>
<tr>
<td>64.8.3</td>
<td>MAC Interrupts</td>
</tr>
<tr>
<td>64.9</td>
<td>Ethernet programming model</td>
</tr>
<tr>
<td>64.9.1</td>
<td>DMA initialization</td>
</tr>
<tr>
<td>64.9.2</td>
<td>MTL initialization</td>
</tr>
<tr>
<td>64.9.3</td>
<td>MAC initialization</td>
</tr>
<tr>
<td>64.9.4</td>
<td>Performing normal receive and transmit operation</td>
</tr>
<tr>
<td>64.9.5</td>
<td>Stopping and starting transmission</td>
</tr>
<tr>
<td>64.9.6</td>
<td>Programming guidelines for switching to new descriptor list in RxDMA</td>
</tr>
<tr>
<td>64.9.7</td>
<td>Programming guidelines for switching the AHB clock frequency</td>
</tr>
<tr>
<td>64.9.8</td>
<td>Programming guidelines for MII link state transitions</td>
</tr>
<tr>
<td>64.9.9</td>
<td>Programming guidelines for IEEE 1588 timestamping</td>
</tr>
<tr>
<td>64.9.10</td>
<td>Programming guidelines for PTP offload feature</td>
</tr>
<tr>
<td>64.9.11</td>
<td>Programming guidelines for Energy Efficient Ethernet (EEE)</td>
</tr>
<tr>
<td>64.9.12</td>
<td>Programming guidelines for flexible pulse-per-second (PPS) output</td>
</tr>
<tr>
<td>64.9.13</td>
<td>Programming guidelines for TSO</td>
</tr>
<tr>
<td>64.9.14</td>
<td>Programming guidelines to perform VLAN filtering on the receiver</td>
</tr>
<tr>
<td>64.10</td>
<td>Descriptors</td>
</tr>
<tr>
<td>64.10.1</td>
<td>Descriptor overview</td>
</tr>
<tr>
<td>64.10.2</td>
<td>Descriptor structure</td>
</tr>
<tr>
<td>64.10.3</td>
<td>Transmit descriptor</td>
</tr>
<tr>
<td>64.10.4</td>
<td>Receive descriptor</td>
</tr>
<tr>
<td>64.11</td>
<td>Ethernet registers</td>
</tr>
<tr>
<td>64.11.1</td>
<td>Ethernet register maps</td>
</tr>
<tr>
<td>64.11.2</td>
<td>Ethernet DMA registers</td>
</tr>
<tr>
<td>64.11.3</td>
<td>Ethernet MTL registers</td>
</tr>
<tr>
<td>64.11.4</td>
<td>Ethernet MAC and MMC registers</td>
</tr>
<tr>
<td>65</td>
<td>HDMI-CEC controller (CEC)</td>
</tr>
<tr>
<td>65.1</td>
<td>HDMI-CEC introduction</td>
</tr>
<tr>
<td>65.2</td>
<td>HDMI-CEC controller main features</td>
</tr>
</tbody>
</table>
65.3 HDMI-CEC functional description ................................. 3538
  65.3.1 HDMI-CEC pin and internal signals ......................... 3538
  65.3.2 HDMI-CEC block diagram .................................. 3539
  65.3.3 Message description ...................................... 3539
  65.3.4 Bit timing ................................................ 3540

65.4 Arbitration ..................................................... 3540
  65.4.1 SFT option bit ............................................ 3542

65.5 Error handling .................................................. 3542
  65.5.1 Bit error ................................................. 3542
  65.5.2 Message error ............................................ 3543
  65.5.3 Bit rising error (BRE) .................................... 3543
  65.5.4 Short bit period error (SBPE) ............................. 3543
  65.5.5 Long bit period error (LBPE) ............................. 3543
  65.5.6 Transmission error detection (TXERR) .................... 3545

65.6 HDMI-CEC interrupts ............................................ 3546

65.7 HDMI-CEC registers ............................................. 3547
  65.7.1 CEC control register (CEC_CR) ............................. 3547
  65.7.2 CEC configuration register (CEC_CFGR) ................... 3548
  65.7.3 CEC Tx data register (CEC_TXDR) ........................ 3550
  65.7.4 CEC Rx data register (CEC_RXDR) ........................ 3550
  65.7.5 CEC interrupt and status register (CEC_ISR) ............. 3550
  65.7.6 CEC interrupt enable register (CEC_IER) ................. 3552
  65.7.7 HDMI-CEC register map .................................... 3554

66 Debug infrastructure ............................................... 3555
  66.1 Introduction .................................................. 3555
  66.2 Debug infrastructure features ............................... 3556
  66.3 Debug infrastructure functional description ............... 3556
    66.3.1 Debug infrastructure block diagram ..................... 3556
    66.3.2 Debug infrastructure powering, clocking and reset ...... 3557
    66.3.3 Security ............................................... 3558
    66.3.4 Debug authentication .................................... 3560
  66.4 Serial-wire and JTAG debug port (SWJ-DP) .................... 3561
    66.4.1 Serial wire debug port .................................. 3562
    66.4.2 JTAG debug port ........................................ 3564
    66.4.3 Debug port registers .................................... 3567
66.4.4 Debug port register map ........................................ 3576
66.5 Access ports ......................................................... 3577
  66.5.1 MEM-AP registers ............................................. 3578
  66.5.2 Access port register map .................................... 3586
66.6 Trace and debug subsystem functional description .............. 3587
  66.6.1 System ROM tables ........................................... 3587
  66.6.2 System ROM registers ......................................... 3591
  66.6.3 System ROM register map and reset values ...................... 3595
  66.6.4 Cross trigger interfaces (CTI) and matrix (CTM) ............. 3597
  66.6.5 CTI registers ................................................ 3600
  66.6.6 CTI register map and reset values .......................... 3614
  66.6.7 Trace funnel (CSTF) .......................................... 3616
  66.6.8 Trace funnel registers ....................................... 3617
  66.6.9 Trace funnel register map and reset values .................... 3626
  66.6.10 Embedded trace FIFO (ETF) .................................. 3628
  66.6.11 ETF registers .............................................. 3630
  66.6.12 ETF register map and reset values .......................... 3647
  66.6.13 Trace port interface unit (TPIU) ............................ 3650
  66.6.14 TPIU registers ............................................ 3651
  66.6.15 TPIU register map and reset values ........................ 3666
66.7 Serial wire output (SWO) ........................................ 3668
  66.7.1 SWO registers .............................................. 3669
  66.7.2 SWO register map and reset values .......................... 3678
66.8 Microcontroller debug unit (DBGMCU) .......................... 3680
  66.8.1 DBGMCU registers .......................................... 3681
  66.8.2 DBGMCU register map and reset values ....................... 3695
66.9 Cortex-M7 debug functional description ......................... 3698
  66.9.1 Cortex-M7 ROM tables ........................................ 3698
  66.9.2 Cortex-M7 CPU ROM registers ................................ 3701
  66.9.3 Cortex-M7 CPU ROM table register map and reset values .... 3705
  66.9.4 Cortex-M7 PPB ROM registers ................................ 3706
  66.9.5 Cortex-M7 PPB ROM table register map and reset values .... 3710
  66.9.6 Cortex-M7 data watchpoint and trace unit (DWT) ............ 3711
  66.9.7 Cortex-M7 DWT registers .................................... 3712
  66.9.8 Cortex-M7 DWT register map and reset values ............... 3721
  66.9.9 Cortex-M7 instrumentation trace macrocell (ITM) ............ 3723
Contents

66.9.10 Cortex-M7 ITM registers .................................................. 3724
66.9.11 Cortex-M7 ITM register map and reset values ...................... 3730
66.9.12 Cortex-M7 breakpoint unit (FPB) ...................................... 3732
66.9.13 Cortex-M7 FPB registers ................................................ 3732
66.9.14 Cortex-M7 FPB register map and reset values ..................... 3737
66.9.15 Cortex-M7 embedded trace macrocell (ETM) ....................... 3738
66.9.16 Cortex-M7 ETM registers .............................................. 3739
66.9.17 Cortex-M7 ETM register map and reset values ..................... 3765
66.9.18 Cortex-M7 cross trigger interface (CTI) ............................ 3769
66.10 References for debug infrastructure ..................................... 3770

67 Device electronic signature ...................................................... 3771
67.1 Unique device ID register (96 bits) ....................................... 3771
67.2 Package data register ........................................................ 3773

68 Important security notice ....................................................... 3774

69 Revision history ................................................................. 3775
List of tables

Table 1. Availability of security features ................................................................. 121
Table 2. Bus-master-to-bus-slave interconnect ....................................................... 122
Table 3. ASIB configuration ................................................................................. 129
Table 4. AMIB configuration ................................................................................. 129
Table 5. AXI interconnect register map and reset values ...................................... 139
Table 6. Memory map and default device memory area attribute ......................... 151
Table 7. Register boundary addresses ................................................................. 153
Table 8. SRAMs structure ..................................................................................... 160
Table 9. SRAM1 / ITCM configurations ................................................................. 160
Table 10. SRAM3 / DTCM configurations ............................................................... 160
Table 11. SRAM4 / ECC configurations ................................................................. 161
Table 12. ITCM/AXI configuration ....................................................................... 161
Table 13. DTCM/AXI configuration ...................................................................... 161
Table 14. ECC/AXI configuration ....................................................................... 161
Table 15. Boot modes ......................................................................................... 162
Table 16. RAMECC internal input/output signals .................................................. 166
Table 17. ECC controller mapping ........................................................................ 167
Table 18. Effect of low-power modes on RAMECC ............................................. 168
Table 19. RAMECC interrupt requests ................................................................. 169
Table 20. RAMECC register map and reset values ............................................... 176
Table 21. Internal tampers in TAMP .................................................................. 184
Table 22. Effect of low-power modes on TAMP .................................................... 186
Table 23. Accelerated cryptographic operations .................................................... 188
Table 24. Product life-cycle states ...................................................................... 191
Table 25. Macros for RSS services ...................................................................... 192
Table 26. RSS lib interface functions ................................................................... 193
Table 27. FLASH internal input/output signals ..................................................... 201
Table 28. Flash memory organization .................................................................. 204
Table 29. FLASH recommended read wait states and programming delays .......... 210
Table 30. Flash memory OTP organization ............................................................ 218
Table 31. read-only public data organization ......................................................... 220
Table 32. Option byte user words organization ..................................................... 225
Table 33. STM32H7Rx/7Sx device lifecycle table .................................................. 229
Table 34. Flash interface register protection summary .......................................... 231
Table 35. Flash security lifecycle definition ............................................................ 232
Table 36. Boot level and HDP area protections (non STiRoT case) ....................... 234
Table 37. Boot level and HDP area protections (STiRoT case) .............................. 234
Table 38. Effect of low-power modes on the embedded flash memory ............... 236
Table 39. Flash interrupt request ...................................................................... 243
Table 40. FLASH register map and reset values ................................................. 274
Table 41. PWR input/output signals connected to package pins or balls .......... 280
Table 42. PWR internal input/output signals ......................................................... 280
Table 43. table wakeup source selection .............................................................. 281
Table 44. Supply configuration control ................................................................. 285
Table 45. Operating mode summary ................................................................. 304
Table 46. Sleep mode ......................................................................................... 311
Table 47. Stop mode operation ......................................................................... 312
Table 48. Stop mode......................................................................................... 313
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>49</td>
<td>Standby and Stop flags</td>
<td>315</td>
</tr>
<tr>
<td>50</td>
<td>Standby mode</td>
<td>315</td>
</tr>
<tr>
<td>51</td>
<td>Low-power modes monitoring pin overview</td>
<td>316</td>
</tr>
<tr>
<td>52</td>
<td>GPIO state according to CPU and domain state</td>
<td>316</td>
</tr>
<tr>
<td>53</td>
<td>Power control register map and reset values</td>
<td>338</td>
</tr>
<tr>
<td>54</td>
<td>RCC input/output signals connected to package pins or balls</td>
<td>342</td>
</tr>
<tr>
<td>55</td>
<td>RCC internal input/output signals</td>
<td>349</td>
</tr>
<tr>
<td>56</td>
<td>Reset coverage summary</td>
<td>349</td>
</tr>
<tr>
<td>57</td>
<td>Reset source identification (RCC_RSR)</td>
<td>350</td>
</tr>
<tr>
<td>58</td>
<td>Oscillator states versus system modes</td>
<td>354</td>
</tr>
<tr>
<td>59</td>
<td>Oscillator states versus system modes</td>
<td>368</td>
</tr>
<tr>
<td>60</td>
<td>VCO frequency and output frequency in integer mode</td>
<td>372</td>
</tr>
<tr>
<td>61</td>
<td>VCO frequency and output frequency in fractional mode</td>
<td>373</td>
</tr>
<tr>
<td>62</td>
<td>Clock protection summary</td>
<td>378</td>
</tr>
<tr>
<td>63</td>
<td>HSIKERON and CSIKERON behavior</td>
<td>380</td>
</tr>
<tr>
<td>64</td>
<td>Peripheral clock distribution summary</td>
<td>381</td>
</tr>
<tr>
<td>65</td>
<td>SDMMC interface clock constraints</td>
<td>388</td>
</tr>
<tr>
<td>66</td>
<td>Ratio between clock timer and pclk</td>
<td>406</td>
</tr>
<tr>
<td>67</td>
<td>Peripheral clock enabling</td>
<td>413</td>
</tr>
<tr>
<td>68</td>
<td>RNG clock enabling</td>
<td>414</td>
</tr>
<tr>
<td>69</td>
<td>Interrupt sources and control</td>
<td>417</td>
</tr>
<tr>
<td>70</td>
<td>Clock configuration examples with PLLs in integer mode</td>
<td>421</td>
</tr>
<tr>
<td>71</td>
<td>Clock configuration examples with PLLs in fractional mode</td>
<td>422</td>
</tr>
<tr>
<td>72</td>
<td>RCC register map and reset values</td>
<td>530</td>
</tr>
<tr>
<td>73</td>
<td>SBS input/output pins</td>
<td>540</td>
</tr>
<tr>
<td>74</td>
<td>SBS internal input/output signals</td>
<td>540</td>
</tr>
<tr>
<td>75</td>
<td>SBS boot control</td>
<td>542</td>
</tr>
<tr>
<td>76</td>
<td>Analog switches recommended configuration</td>
<td>543</td>
</tr>
<tr>
<td>77</td>
<td>SBS register map and reset values</td>
<td>558</td>
</tr>
<tr>
<td>78</td>
<td>CRS features</td>
<td>560</td>
</tr>
<tr>
<td>79</td>
<td>CRS internal input/output signals</td>
<td>561</td>
</tr>
<tr>
<td>80</td>
<td>CRS internal input/output signals</td>
<td>562</td>
</tr>
<tr>
<td>81</td>
<td>Effect of low-power modes on CRS</td>
<td>565</td>
</tr>
<tr>
<td>82</td>
<td>Interrupt control bits</td>
<td>565</td>
</tr>
<tr>
<td>83</td>
<td>Programmed GPDMA1 register map and reset values</td>
<td>570</td>
</tr>
<tr>
<td>84</td>
<td>CRS register map and reset values</td>
<td>572</td>
</tr>
<tr>
<td>85</td>
<td>Port bit configuration</td>
<td>572</td>
</tr>
<tr>
<td>86</td>
<td>GPIO register map and reset values</td>
<td>588</td>
</tr>
<tr>
<td>87</td>
<td>Peripheral interconnect matrix</td>
<td>590</td>
</tr>
<tr>
<td>88</td>
<td>GPDMA1 channel implementation</td>
<td>601</td>
</tr>
<tr>
<td>89</td>
<td>GPDMA1 in low-power modes</td>
<td>602</td>
</tr>
<tr>
<td>90</td>
<td>Programmed GPDMA1 request</td>
<td>602</td>
</tr>
<tr>
<td>91</td>
<td>Programmed GPDMA1 request as a block request</td>
<td>605</td>
</tr>
<tr>
<td>92</td>
<td>GPDMA channel with peripheral early termination</td>
<td>606</td>
</tr>
<tr>
<td>93</td>
<td>Programmed GPDMA request with peripheral early termination</td>
<td>606</td>
</tr>
<tr>
<td>94</td>
<td>Programmed GPDMA trigger</td>
<td>606</td>
</tr>
<tr>
<td>95</td>
<td>Programmed GPDMA source/destination burst</td>
<td>628</td>
</tr>
<tr>
<td>96</td>
<td>Programmed data handling</td>
<td>633</td>
</tr>
<tr>
<td>97</td>
<td>Effect of low-power modes on GPDMA</td>
<td>646</td>
</tr>
<tr>
<td>98</td>
<td>GPDMA interrupt requests</td>
<td>647</td>
</tr>
<tr>
<td>99</td>
<td>GPDMA register map and reset values</td>
<td>673</td>
</tr>
<tr>
<td>100</td>
<td>Implementation of HPDMA1 channels</td>
<td>676</td>
</tr>
</tbody>
</table>
Table 121. Supported color mode in output
Table 122. Supported color mode in input
Table 123. Data order in memory
Table 124. Standard data order in memory
Table 125. Output FIFO byte reordering steps
Table 126. DMA2D interrupt requests
Table 127. DMA2D internal signals
Table 140. Graphic timer interrupt requests
Table 141. GFXTIM register map and reset values
Table 142. NVIC
Table 143. EXTI event input configurations and register control
Table 144. EXTI event input mapping
Table 145. Masking functionality
Table 146. Extended interrupt and event controller register map and reset values
Table 147. CRC internal input/output signals
Table 148. CRC register map and reset values
Table 149. CORDIC functions
Table 150. Cosine parameters
Table 151. Sine parameters

Table 101. HPDMA1 in low-power modes
Table 102. Programmed HPDMA1 request
Table 103. Programmed HPDMA1 request as a block request
Table 104. HPDMA channel with peripheral early termination
Table 105. Programmed HPDMA request with peripheral early termination
Table 106. Programmed HPDMA1 trigger
Table 107. Programmed HPDMA source/destination burst
Table 108. Programmed data handling
Table 109. Effect of low-power modes on HPDMA
Table 110. HPDMA interrupt requests
Table 111. HPDMA register map and reset values
Table 112. GFXMMU internal input/output signals
Table 113. GFXMMU interrupt requests
Table 114. GFXMMU register map and reset values
Table 115. DMA2D internal signals
Table 116. DMA2D trigger interconnections
Table 117. Supported color mode in input
Table 118. Data order in memory
Table 119. Alpha mode configuration
Table 120. Supported CLUT color mode
Table 121. CLUT data order in memory
Table 122. Supported color mode in output
Table 123. Data order in memory
Table 124. Standard data order in memory
Table 125. Output FIFO byte reordering steps
Table 126. DMA2D interrupt requests
Table 127. DMA2D internal signals
Table 128. DMA2D register map and reset values
Table 129. GPU2D internal input/output signals
Table 130. GPU2D trigger connections
Table 131. ICACHE features
Table 132. TAG memory dimensioning parameters
Table 133. TAG memory dimensioning parameters for direct-mapped cache mode
Table 134. ICACHE cacheability for AHB transaction
Table 135. ICACHE interrupts
Table 136. ICACHE register map and reset values
Table 137. GFXTIM input/output pins
Table 138. GFXTIM internal signals
Table 139. GFXTIM trigger interconnections
Table 140. Graphic timer interrupt requests
Table 141. GFXTIM register map and reset values
Table 142. NVIC
Table 143. EXTI event input configurations and register control
Table 144. EXTI event input mapping
Table 145. Masking functionality
Table 146. Extended interrupt and event controller register map and reset values
Table 147. CRC internal input/output signals
Table 148. CRC register map and reset values
Table 149. CORDIC functions
Table 150. Cosine parameters
Table 151. Sine parameters
Table 177. 16-bit multiplexed I/O NOR flash memory .......................... 926
Table 176. Non-multiplexed I/O NOR flash memory ......................... 925
Table 175. Programmable NOR/PSRAM access parameters .................. 925
Table 174. SDRAM address mapping with 32-bit data bus width ............... 923
Table 173. SDRAM address mapping ............................................ 921
Table 172. NAND bank selection .................................................. 920
Table 171. NAND memory mapping and timing registers ....................... 920
Table 170. NOR flash/PSRAM: Example of supported memories and transactions ................................. 927
Table 169. NAND memory mapping and timing registers ....................... 920
Table 168. NAND memory mapping and timing registers ....................... 920
Table 167. NOR/PSRAM External memory address .............................. 919
Table 166. NOR/PSRAM bank selection .......................................... 919
Table 165. FMC bank mapping options ......................................... 916
Table 164. FMC pins ............................................................... 912
Table 163. CORDIC register map and reset value .............................. 912
Table 162. Precision vs. number of iterations .................................. 905
Table 161. Square root scaling factors and corresponding ranges .......... 902
Table 160. Square root parameters .............................................. 902
Table 159. Natural log scaling factors and corresponding ranges .......... 901
Table 158. Natural logarithm parameters ...................................... 901
Table 157. Hyperbolic arctangent parameters ................................ 900
Table 156. Hyperbolic cosine parameters .................................... 899
Table 155. Hyperbolic sine parameters ....................................... 899
Table 154. Arctangent parameters ............................................. 898
Table 153. Modulus parameters ................................................... 897
Table 152. Phase parameters ....................................................... 897
Table 203. 16-bit NAND flash memory ................................................................. 964
Table 204. Supported memories and transactions ................................................. 964
Table 205. ECC result relevant bits .................................................................. 974
Table 206. SDRAM signals ............................................................................. 975
Table 207. FMC register map ......................................................................... 992
Table 208. XSPI implementation ..................................................................... 996
Table 209. XSPI input/output pins ................................................................. 1001
Table 210. XSPI internal signals .................................................................... 1001
Table 211. Command/address phase description .............................................. 1012
Table 212. ADC features .............................................................................. 1082
Table 213. Memory location of the temperature sensor calibration values .......... 1080
Table 214. DLYB interconnection ................................................................... 1074
Table 215. XSPI interrupt requests .................................................................. 1031
Table 216. XSPI register map and reset values ............................................... 1058
Table 217. Address alignment cases ............................................................... 1033
Table 218. Configuration for master ADC, 0x100 for slave ADC) ................... 1023
Table 219. XSPI register map and reset values ............................................... 1064
Table 220. DLYB internal input/output signals ............................................... 1023
Table 221. DLYB interconnection ................................................................... 1074
Table 222. OctaRAM command address bit assignment (based on 64 Mb OctaRAM) ................................................................. 1023
Table 223. DLYB register map and reset values ............................................... 1074
Table 224. ADC features .............................................................................. 1077
Table 225. Memory location of the temperature sensor calibration values .......... 1080
Table 226. Memory location of the internal reference voltage sensor calibration value .................................................................
Table 227. ADC input/output pins ................................................................. 1082
Table 228. ADC internal input/output signals .................................................. 1082
Table 229. ADC interconnection ................................................................... 1082
Table 230. Configuring the trigger polarity for regular external triggers .......... 1102
Table 231. Configuring the trigger polarity for injected external triggers ......... 1103
Table 232. TSAR timings depending on resolution ............................................ 1115
Table 233. Offset computation versus data resolution ....................................... 1118
Table 234. Analog watchdog channel selection .............................................. 1130
Table 235. Analog watchdog 1 comparison .................................................... 1131
Table 236. Analog watchdog 2 and 3 comparison ........................................... 1132
Table 237. Maximum output results versus N and M (gray cells indicate truncation) ................................................................. 1135
Table 238. Oversampler operating modes summary ......................................... 1140
Table 239. Effect of low-power modes on the ADC ........................................ 1159
Table 240. ADC interrupts ........................................................................... 1160
Table 241. DELAY bits versus ADC resolution ............................................. 1195
Table 242. ADC global register map ............................................................... 1199
Table 243. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) ................................................................. 1199
Table 244. ADC register map and reset values (master and slave ADC common registers) ................................................................. 1201
Table 245. DTS internal input/output signals ................................................... 1204
Table 246. Sampling time configuration .......................................................... 1207
Table 247. Trigger configuration .................................................................... 1208
Table 248. Temperature sensor behavior in low-power modes ....................... 1210
Table 249. Interrupt control bits .................................................................... 1211
Table 250. DTS register map and reset values ............................................... 1219
Table 251. VREFBUF typical values ................................................................. 1220
Table 252. VREF buffer modes ................................................................. 1221
Table 253. VREFBUF register map and reset values. .............................. 1223
Table 254. ADF features ................................................................. 1225
Table 255. ADF external pins ................................................................. 1226
Table 256. ADF internal signals ............................................................. 1226
Table 257. ADF trigger connections ...................................................... 1227
Table 258. Control of the common clock generation .............................. 1233
Table 259. Clock constraints with respect to the incoming stream .......... 1234
Table 260. Data size according to CIC order and CIC decimation values .... 1239
Table 261. Possible gain values .............................................................. 1240
Table 262. Recommended maximum gain values versus CIC decimation ratios . 1242
Table 263. Most common microphone settings .................................... 1243
Table 264. HPF 3 dB cut-off frequencies examples ............................... 1245
Table 265. ANSLP values versus FRSIZE and sampling rates ................ 1258
Table 266. Threshold values according SNTHR ...................................... 1259
Table 267. Register protection summary .............................................. 1265
Table 268. Effect of low-power modes on ADF ...................................... 1266
Table 269. ADF interrupt requests ....................................................... 1267
Table 270. Examples of ADF settings for microphone capture ............... 1268
Table 271. Programming sequence (CIC4) ............................................. 1269
Table 272. Programming sequence (CIC5) ............................................. 1270
Table 273. Output signal levels ............................................................ 1275
Table 274. ADF register map and reset values ...................................... 1297
Table 275. Available pipeline ............................................................... 1300
Table 276. Glossary ................................................................. 1301
Table 277. DCMIPP input/output pads .................................................. 1303
Table 278. DCMIPP input/output pins ................................................... 1304
Table 279. DCMIPP clocks ................................................................. 1304
Table 280. DCMIPP resets ................................................................. 1304
Table 281. Parallel interface maximum resolution (80 MHz) ................. 1305
Table 282. Supported pixel formats ..................................................... 1306
Table 283. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function .... 1309
Table 284. DCMIPP_P0PPCR bit function ............................................ 1309
Table 285. DCMIPP_PxFCTCR bit function ........................................... 1311
Table 286. DCMIPP_P0PPCR bit function ............................................ 1313
Table 287. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function .... 1319
Table 288. DCMIPP_PxFCTCR bit function ........................................... 1320
Table 289. Parallel interface input pixel formats ................................ 1321
Table 290. Correspondence between index and DCMIPP_PRCR register values .............................................. 1322
Table 291. Parallel interface input pixel formats ................................ 1323
Table 292. Dump pipe OUTPUT pixel formats ..................................... 1323
Table 293. Shadow and physical registers ......................................... 1326
Table 294. DCMIPP low power modes ............................................... 1327
Table 295. DCMIPP interrupts ............................................................. 1330
Table 296. Event connection ............................................................... 1330
Table 297. DCMIPP registers organization ....................................... 1331
Table 298. DCMIPP register map and reset values ................................ 1358
Table 299. PSSI input/output pins ....................................................... 1362
Table 300. PSSI internal input/output signals ..................................... 1363
Table 301. Positioning of captured data bytes in 32-bit words (8-bit width) 1364
Table 302. Positioning of captured data bytes in 32-bit words (16-bit width) 1365
RM0477
Table 303.
Table 304.
Table 305.
Table 306.
Table 307.
Table 308.
Table 309.
Table 310.
Table 311.
Table 312.
Table 313.
Table 314.
Table 315.
Table 316.
Table 317.
Table 318.
Table 319.
Table 320.
Table 321.
Table 322.
Table 323.
Table 324.
Table 325.
Table 326.
Table 327.
Table 328.
Table 329.
Table 330.
Table 331.
Table 332.
Table 333.
Table 334.
Table 335.
Table 336.
Table 337.
Table 338.
Table 339.
Table 340.
Table 341.
Table 342.
Table 343.
Table 344.
Table 345.
Table 346.
Table 347.
Table 348.
Table 349.
Table 350.
Table 351.
Table 352.
Table 353.
Table 354.

List of tables
PSSI interrupt requests. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1368
PSSI register map and reset values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1374
LTDC external pins . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1376
LTDC internal signals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1377
LTDC trigger interconnection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1377
Clock domain for each register . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1377
LTDC register access and update durations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1378
Pixel data mapping versus color format . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1382
LTDC interrupt requests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1386
LTDC register map and reset values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1403
JPEG internal signals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1407
JPEG trigger connections . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1408
JPEG codec interrupt requests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1413
JPEG codec register map and reset values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1426
RNG internal input/output signals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1429
RNG interrupt requests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1437
RNG initialization times. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1438
RNG configurations . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1439
Configuration selection . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1439
RNG register map and reset map . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1445
CRYP versus SAES features . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1447
SAES internal input/output signals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1448
SAES approved symmetric key functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1449
Counter mode initialization vector definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1459
Initialization of IV registers in GCM mode. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1462
GCM last block definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1462
Initialization of IV registers in CCM mode . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1468
AES data swapping example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1477
Key endianness in SAES_KEYRx registers (128/256-bit keys) . . . . . . . . . . . . . . . . . . . 1479
IVI bitfield spread over SAES_IVRx registers . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1480
SAES interrupt requests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1482
Processing latency for ECB, CBC and CTR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1483
Processing latency for GCM and CCM (in SAES kernel clock cycles) . . . . . . . . . . . . . . 1484
SAES register map and reset values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1497
CRYP versus SAES features. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1501
CRYP internal input/output signals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1503
CRYP approved AES symmetric key functions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1503
Counter mode initialization vector definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1514
GCM mode IVI registers initialization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1517
GCM last block definition . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1517
CCM mode IVI registers initialization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1524
AES data swapping example . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1529
Key endianness in CRYP_KxR/LR registers (128/192/256-bit keys) . . . . . . . . . . . . . . . 1531
Initialization vector endianness in CRYP_IVxR registers (AES) . . . . . . . . . . . . . . . . . . . 1531
CRYP interrupt requests . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1533
Processing latency for ECB, CBC and CTR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1534
Processing latency for GCM and CCM (in clock cycles). . . . . . . . . . . . . . . . . . . . . . . . . 1535
CRYP register map and reset values . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1548
HASH internal input/output signals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1551
Information on supported hash algorithms . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1552
Hash processor outputs . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1555
Processing time (in clock cycle) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1561

RM0477 Rev 6

89/3791
98


List of tables

Table 355. HASH interrupt requests ................................................. 1562
Table 356. HASH1 register map and reset values ............................. 1570
Table 357. MCE implementation .................................................. 1573
Table 358. MCE internal input/output signals ................................ 1574
Table 359. MCE block cipher latencies ......................................... 1578
Table 360. MCE stream cipher latencies ....................................... 1579
Table 361. MCE interrupt requests ............................................. 1580
Table 362. MCE register map and reset values ............................... 1594
Table 363. Internal input/output signals ...................................... 1597
Table 364. PKA integer arithmetic functions list ............................ 1598
Table 365. PKA prime field (Fp) elliptic curve functions list ............. 1599
Table 366. Montgomery parameter computation .............................. 1605
Table 367. Modular addition .................................................... 1606
Table 368. Modular subtraction .................................................. 1606
Table 369. Montgomery multiplication ....................................... 1607
Table 370. Modular exponentiation (normal mode) ............................ 1608
Table 371. Modular exponentiation (fast mode) ............................... 1608
Table 372. Modular exponentiation (protected mode) ......................... 1609
Table 373. Modular inversion .................................................... 1609
Table 374. Modular reduction .................................................... 1610
Table 375. Arithmetic addition .................................................. 1610
Table 376. Arithmetic subtraction ............................................. 1610
Table 377. Arithmetic multiplication ......................................... 1611
Table 378. Arithmetic comparison ............................................ 1611
Table 379. CRT exponentiation .................................................. 1612
Table 380. Point on elliptic curve Fp check .................................. 1613
Table 381. ECC Fp scalar multiplication ...................................... 1613
Table 382. ECDSA sign - Inputs .................................................. 1615
Table 383. ECDSA sign - Outputs ............................................... 1615
Table 384. Extended ECDSA sign - Extra outputs ............................ 1616
Table 385. ECDSA verification - Inputs ....................................... 1616
Table 386. ECDSA verification - Outputs .................................... 1617
Table 387. ECC complete addition ............................................ 1617
Table 388. ECC double base ladder ............................................ 1618
Table 389. ECC projective to affine ........................................... 1619
Table 390. Family of supported curves for ECC operations ................. 1620
Table 391. Modular exponentiation ............................................ 1621
Table 392. ECC scalar multiplication ........................................ 1621
Table 393. ECDSA signature average computation time ...................... 1622
Table 394. ECDSA verification average computation times ................. 1622
Table 395. ECC double base ladder average computation times .......... 1622
Table 396. ECC projective to affine average computation times .......... 1622
Table 397. ECC complete addition average computation times .......... 1622
Table 398. Point on elliptic curve Fp check average computation times 1622
Table 399. Montgomery parameters average computation times .......... 1623
Table 400. PKA interrupt requests ............................................. 1623
Table 401. PKA register map and reset values ............................... 1628
Table 402. TIM input/output pins .............................................. 1632
Table 403. TIM internal input/output signals ................................ 1632
Table 404. Interconnect to the tim_ti1 input multiplexer ................... 1633
Table 405. Interconnect to the tim_ti2 input multiplexer ................... 1633
Table 406. Interconnect to the tim_ti3 input multiplexer ................... 1634
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>407</td>
<td>Interconnect to the tim_ti4 input multiplexer</td>
<td>1634</td>
</tr>
<tr>
<td>408</td>
<td>Internal trigger connection</td>
<td>1634</td>
</tr>
<tr>
<td>409</td>
<td>Interconnect to the tim_etr input multiplexer</td>
<td>1634</td>
</tr>
<tr>
<td>410</td>
<td>Timer break interconnect</td>
<td>1635</td>
</tr>
<tr>
<td>411</td>
<td>Timer break2 interconnect</td>
<td>1635</td>
</tr>
<tr>
<td>412</td>
<td>System break interconnect</td>
<td>1635</td>
</tr>
<tr>
<td>413</td>
<td>CCR and ARR register change dithering pattern</td>
<td>1668</td>
</tr>
<tr>
<td>414</td>
<td>CCR register change dithering pattern in center-aligned PWM mode</td>
<td>1669</td>
</tr>
<tr>
<td>415</td>
<td>Behavior of timer outputs versus tim_brk/tim_brk2 inputs</td>
<td>1681</td>
</tr>
<tr>
<td>416</td>
<td>Break protection disarming conditions</td>
<td>1683</td>
</tr>
<tr>
<td>417</td>
<td>Counting direction versus encoder signals (CC1P = CC2P = 0)</td>
<td>1691</td>
</tr>
<tr>
<td>418</td>
<td>Counting direction versus encoder signals and polarity settings</td>
<td>1695</td>
</tr>
<tr>
<td>419</td>
<td>DMA request</td>
<td>1716</td>
</tr>
<tr>
<td>420</td>
<td>Effect of low-power modes on TIM1</td>
<td>1717</td>
</tr>
<tr>
<td>421</td>
<td>Interrupt requests</td>
<td>1717</td>
</tr>
<tr>
<td>422</td>
<td>Output control bits for complementary tim.ocx and tim.ocxn channels</td>
<td>1744</td>
</tr>
<tr>
<td></td>
<td>with break feature</td>
<td></td>
</tr>
<tr>
<td>423</td>
<td>TIM1 register map and reset values</td>
<td>1767</td>
</tr>
<tr>
<td>424</td>
<td>STM32H7Rx/7Sx general purpose timers</td>
<td>1771</td>
</tr>
<tr>
<td>425</td>
<td>TIM input/output pins</td>
<td>1773</td>
</tr>
<tr>
<td>426</td>
<td>TIM internal input/output signals</td>
<td>1773</td>
</tr>
<tr>
<td>427</td>
<td>Interconnect to the tim_ti1 input multiplexer</td>
<td>1774</td>
</tr>
<tr>
<td>428</td>
<td>Interconnect to the tim_ti2 input multiplexer</td>
<td>1774</td>
</tr>
<tr>
<td>429</td>
<td>Interconnect to the tim_ti3 input multiplexer</td>
<td>1774</td>
</tr>
<tr>
<td>430</td>
<td>Interconnect to the tim_ti4 input multiplexer</td>
<td>1775</td>
</tr>
<tr>
<td>431</td>
<td>TIMx internal trigger connection</td>
<td>1775</td>
</tr>
<tr>
<td>432</td>
<td>Interconnect to the tim_etr input multiplexer</td>
<td>1775</td>
</tr>
<tr>
<td>433</td>
<td>CCR and ARR register change dithering pattern</td>
<td>1807</td>
</tr>
<tr>
<td>434</td>
<td>CCR register change dithering pattern in center-aligned PWM mode</td>
<td>1808</td>
</tr>
<tr>
<td>435</td>
<td>Counting direction versus encoder signals(CC1P = CC2P = 0)</td>
<td>1817</td>
</tr>
<tr>
<td>436</td>
<td>Counting direction versus encoder signals and polarity settings</td>
<td>1822</td>
</tr>
<tr>
<td>437</td>
<td>DMA request</td>
<td>1846</td>
</tr>
<tr>
<td>438</td>
<td>Effect of low-power modes on TIM2/TIM3/TIM4/TIM5</td>
<td>1846</td>
</tr>
<tr>
<td>439</td>
<td>Interrupt requests</td>
<td>1847</td>
</tr>
<tr>
<td>440</td>
<td>Output control bit for standard tim_ocx channels</td>
<td>1847</td>
</tr>
<tr>
<td>441</td>
<td>TIM2/TIM3/TIM4/TIM5 register map and reset values</td>
<td>1881</td>
</tr>
<tr>
<td>442</td>
<td>TIM internal input/output signals</td>
<td>1885</td>
</tr>
<tr>
<td>443</td>
<td>TIMx_ARR register change dithering pattern</td>
<td>1894</td>
</tr>
<tr>
<td>444</td>
<td>DMA request</td>
<td>1895</td>
</tr>
<tr>
<td>445</td>
<td>Effect of low-power modes on TIM6/TIM7</td>
<td>1895</td>
</tr>
<tr>
<td>446</td>
<td>Interrupt request</td>
<td>1895</td>
</tr>
<tr>
<td>447</td>
<td>TIMx register map and reset values</td>
<td>1901</td>
</tr>
<tr>
<td>448</td>
<td>TIM input/output pins</td>
<td>1905</td>
</tr>
<tr>
<td>449</td>
<td>TIM internal input/output signals</td>
<td>1905</td>
</tr>
<tr>
<td>450</td>
<td>Interconnect to the tim_ti1 input multiplexer</td>
<td>1905</td>
</tr>
<tr>
<td>451</td>
<td>Interconnect to the tim_ti2 input multiplexer</td>
<td>1906</td>
</tr>
<tr>
<td>452</td>
<td>TIMx internal trigger connection</td>
<td>1906</td>
</tr>
<tr>
<td>453</td>
<td>CCR and ARR register change dithering pattern</td>
<td>1924</td>
</tr>
<tr>
<td>454</td>
<td>Effect of low-power modes on TIM9/TIM12/TIM13/TIM14</td>
<td>1934</td>
</tr>
<tr>
<td>455</td>
<td>Interrupt requests</td>
<td>1934</td>
</tr>
<tr>
<td>456</td>
<td>Output control bit for standard tim_ocx channels</td>
<td>1947</td>
</tr>
<tr>
<td>457</td>
<td>TIMx register map and reset values (x = 9, 12)</td>
<td>1951</td>
</tr>
</tbody>
</table>
Table 458. Output control bit for standard tim_ocx channels .......................................................... 1960
Table 459. TIM13/TIM14 register map and reset values ................................................................. 1963
Table 460. TIM input/output pins ................................................................................................... 1968
Table 461. TIM internal input/output signals .................................................................................. 1969
Table 462. Interconnect to the tim_ti1 input multiplexer ................................................................. 1969
Table 463. Interconnect to the tim_ti2 input multiplexer ................................................................. 1970
Table 464. TIMx internal trigger connection ..................................................................................... 1970
Table 465. System break interconnect ............................................................................................. 1971
Table 466. CCR and ARR register change dithering pattern ............................................................ 1991
Table 467. Break protection disarming conditions ......................................................................... 2000
Table 468. DMA request ................................................................................................................ 2011
Table 469. Effects of low-power modes on TIM15/TIM16/TIM17 .................................................... 2012
Table 470. Interrupt requests .......................................................................................................... 2012
Table 471. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) ................................................................. 2028
Table 472. TIM15 register map and reset values .............................................................................. 2041
Table 473. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) ................................................................. 2054
Table 474. TIM16/TIM17 register map and reset values ................................................................. 2068
Table 475. STM32H7Rx/7Sx LPTIM features .................................................................................... 2071
Table 476. LPTIM1/2/3 input/output pins ........................................................................................ 2073
Table 477. LPTIM4/5 input/output pins ............................................................................................ 2073
Table 478. LPTIM1/2/3 internal signals ............................................................................................ 2074
Table 479. LPTIM1/2/3 internal signals ............................................................................................ 2074
Table 480. LPTIM1/2/3 input capture 2 connection ......................................................................... 2075
Table 481. LPTIM1/2/3 input capture 2 connection ......................................................................... 2075
Table 482. LPTIM1/2/3 input 1 connection ....................................................................................... 2075
Table 483. LPTIM1/2/3 input 2 connection ....................................................................................... 2075
Table 484. LPTIM1/2/3 input capture 1 connection ......................................................................... 2075
Table 485. LPTIM1 input capture 2 connection ............................................................................... 2076
Table 486. LPTIM2 input capture 2 connection ............................................................................... 2076
Table 487. LPTIM3 input capture 2 connection ............................................................................... 2076
Table 488. Prescaler division ratios ................................................................................................. 2077
Table 489. Encoder counting scenarios ............................................................................................ 2084
Table 490. Input capture Glitch filter latency (in counter step unit) .................................................... 2088
Table 491. Interrupt events ............................................................................................................. 2093
Table 492. LPTIM register map and reset values .............................................................................. 2117
Table 493. WWDG features ............................................................................................................ 2120
Table 494. WWDG internal input/output signals ............................................................................. 2121
Table 495. WWDG register map and reset values .......................................................................... 2126
Table 496. IWDG features .............................................................................................................. 2127
Table 497. IWDG internal input/output signals ............................................................................... 2129
Table 498. Effect of low power modes on IWDG ............................................................................. 2134
Table 499. IWDG interrupt request ................................................................................................ 2136
Table 500. IWDG register map and reset values .............................................................................. 2136
Table 501. RTC input/output pins .................................................................................................... 2142
Table 502. RTC internal input/output signals .................................................................................... 2145
Table 503. RTC interconnection ...................................................................................................... 2145
Table 504. RTC pin PC13 configuration ........................................................................................... 2146
Table 505. RTC_OUT mapping ......................................................................................................... 2149
Table 506. Effect of low-power modes on RTC ............................................................................... 2162
Table 507. RTC pins functionality over modes ................................................................................. 2163
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>508</td>
<td>Interrupt requests</td>
<td>2163</td>
</tr>
<tr>
<td>509</td>
<td>RTC register map and reset values</td>
<td>2190</td>
</tr>
<tr>
<td>510</td>
<td>TAMPS input/output pins</td>
<td>2194</td>
</tr>
<tr>
<td>511</td>
<td>TAMPS internal input/output signals</td>
<td>2194</td>
</tr>
<tr>
<td>512</td>
<td>TAMPS interconnection</td>
<td>2195</td>
</tr>
<tr>
<td>513</td>
<td>Device resource x tamper protection</td>
<td>2200</td>
</tr>
<tr>
<td>514</td>
<td>Active tamper output change period</td>
<td>2202</td>
</tr>
<tr>
<td>515</td>
<td>Minimum ATPER value</td>
<td>2203</td>
</tr>
<tr>
<td>516</td>
<td>Active tamper filtered pulse duration</td>
<td>2204</td>
</tr>
<tr>
<td>517</td>
<td>Effect of low-power modes on TAMPS</td>
<td>2205</td>
</tr>
<tr>
<td>518</td>
<td>TAMPS pins functionality over modes</td>
<td>2205</td>
</tr>
<tr>
<td>519</td>
<td>Interrupt requests</td>
<td>2206</td>
</tr>
<tr>
<td>520</td>
<td>TAMPS register map and reset values</td>
<td>2232</td>
</tr>
<tr>
<td>521</td>
<td>I2C implementation</td>
<td>2235</td>
</tr>
<tr>
<td>522</td>
<td>I2C input/output pins</td>
<td>2236</td>
</tr>
<tr>
<td>523</td>
<td>I2C internal input/output signals</td>
<td>2237</td>
</tr>
<tr>
<td>524</td>
<td>Comparison of analog and digital filters</td>
<td>2239</td>
</tr>
<tr>
<td>525</td>
<td>I2C-bus and SMBus specification data setup and hold times</td>
<td>2241</td>
</tr>
<tr>
<td>526</td>
<td>I2C configuration</td>
<td>2245</td>
</tr>
<tr>
<td>527</td>
<td>I2C-bus and SMBus specification clock timings</td>
<td>2256</td>
</tr>
<tr>
<td>528</td>
<td>Timing settings for $f_{I2CCCLK}$ of 8 MHz</td>
<td>2266</td>
</tr>
<tr>
<td>529</td>
<td>Timing settings for $f_{I2CCCLK}$ of 16 MHz</td>
<td>2266</td>
</tr>
<tr>
<td>530</td>
<td>SMBus timeout specifications</td>
<td>2268</td>
</tr>
<tr>
<td>531</td>
<td>SMBus with PEC configuration</td>
<td>2270</td>
</tr>
<tr>
<td>532</td>
<td>TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms</td>
<td>2271</td>
</tr>
<tr>
<td>533</td>
<td>TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms</td>
<td>2271</td>
</tr>
<tr>
<td>534</td>
<td>TIMEOUTA[11:0] for maximum $t_{IDLE}$ of 50 µs</td>
<td>2271</td>
</tr>
<tr>
<td>535</td>
<td>Effect of low-power modes to I2C</td>
<td>2281</td>
</tr>
<tr>
<td>536</td>
<td>I2C interrupt requests</td>
<td>2281</td>
</tr>
<tr>
<td>537</td>
<td>I2C register map and reset values</td>
<td>2297</td>
</tr>
<tr>
<td>538</td>
<td>I2C wake-up</td>
<td>2300</td>
</tr>
<tr>
<td>539</td>
<td>I2C FIFOs implementation</td>
<td>2300</td>
</tr>
<tr>
<td>540</td>
<td>I2C interrupt(s)</td>
<td>2300</td>
</tr>
<tr>
<td>541</td>
<td>I2C peripheral controller/target features versus MIPI v1.1</td>
<td>2301</td>
</tr>
<tr>
<td>542</td>
<td>I2C input/output pins</td>
<td>2302</td>
</tr>
<tr>
<td>543</td>
<td>I2C internal input/output signals</td>
<td>2302</td>
</tr>
<tr>
<td>544</td>
<td>I2C register usage</td>
<td>2314</td>
</tr>
<tr>
<td>545</td>
<td>I2C registers/fields usage versus controller state</td>
<td>2315</td>
</tr>
<tr>
<td>546</td>
<td>I2C registers/fields usage versus target state</td>
<td>2318</td>
</tr>
<tr>
<td>547</td>
<td>List of supported I2C CCCs, as controller/target</td>
<td>2321</td>
</tr>
<tr>
<td>548</td>
<td>I2C controller error management</td>
<td>2352</td>
</tr>
<tr>
<td>549</td>
<td>I2C target error management</td>
<td>2354</td>
</tr>
<tr>
<td>550</td>
<td>Effect of low-power modes</td>
<td>2358</td>
</tr>
<tr>
<td>551</td>
<td>I2C interrupt requests</td>
<td>2359</td>
</tr>
<tr>
<td>552</td>
<td>I2C register map and reset values</td>
<td>2403</td>
</tr>
<tr>
<td>553</td>
<td>Instance implementation on STM32N7Rx/7Sx</td>
<td>2408</td>
</tr>
<tr>
<td>554</td>
<td>USART/LPUART features</td>
<td>2409</td>
</tr>
<tr>
<td>555</td>
<td>USART/UART input/output pins</td>
<td>2411</td>
</tr>
<tr>
<td>556</td>
<td>USART internal input/output signals</td>
<td>2412</td>
</tr>
<tr>
<td>557</td>
<td>Noise detection from sampled data</td>
<td>2424</td>
</tr>
<tr>
<td>558</td>
<td>Tolerance of the USART receiver when BRR[3:0] = 0000.</td>
<td>2428</td>
</tr>
<tr>
<td>559</td>
<td>Tolerance of the USART receiver when BRR[3:0] is different from 0000.</td>
<td>2428</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>560</td>
<td>USART frame formats</td>
<td></td>
</tr>
<tr>
<td>561</td>
<td>Effect of low-power modes on the USART</td>
<td></td>
</tr>
<tr>
<td>562</td>
<td>USART interrupt requests</td>
<td></td>
</tr>
<tr>
<td>563</td>
<td>USART register map and reset values</td>
<td></td>
</tr>
<tr>
<td>564</td>
<td>Instance implementation on STM32H7Rx/7Sx</td>
<td></td>
</tr>
<tr>
<td>565</td>
<td>USART/LPUART features</td>
<td></td>
</tr>
<tr>
<td>566</td>
<td>LPUART input/output pins</td>
<td></td>
</tr>
<tr>
<td>567</td>
<td>LPUART internal input/output signals</td>
<td></td>
</tr>
<tr>
<td>568</td>
<td>Error calculation for programmed baud rates at luart_ker_ck_pres= 32.768 kHz</td>
<td></td>
</tr>
<tr>
<td>569</td>
<td>Error calculation for programmed baud rates at fCK = 100 MHz</td>
<td></td>
</tr>
<tr>
<td>570</td>
<td>Tolerance of the LPUART receiver</td>
<td></td>
</tr>
<tr>
<td>571</td>
<td>Effect of low-power modes on the LPUART</td>
<td></td>
</tr>
<tr>
<td>572</td>
<td>LPUART interrupt requests</td>
<td></td>
</tr>
<tr>
<td>573</td>
<td>SPI features</td>
<td></td>
</tr>
<tr>
<td>574</td>
<td>SPI/I2S input/output pins</td>
<td></td>
</tr>
<tr>
<td>575</td>
<td>SPI internal input/output signals</td>
<td></td>
</tr>
<tr>
<td>576</td>
<td>Effect of low-power modes on the SPI</td>
<td></td>
</tr>
<tr>
<td>577</td>
<td>SPI wake-up and interrupt requests</td>
<td></td>
</tr>
<tr>
<td>578</td>
<td>Bitfields usable in PCM/I2S mode</td>
<td></td>
</tr>
<tr>
<td>579</td>
<td>WS and CK level before SPI/I2S is enabled when AFCNTR = 1</td>
<td></td>
</tr>
<tr>
<td>580</td>
<td>Serial data line swapping</td>
<td></td>
</tr>
<tr>
<td>581</td>
<td>CLKGEN programming examples for usual I2S frequencies</td>
<td></td>
</tr>
<tr>
<td>582</td>
<td>I2S interrupt requests</td>
<td></td>
</tr>
<tr>
<td>583</td>
<td>SPI register map and reset values</td>
<td></td>
</tr>
<tr>
<td>584</td>
<td>STM32H7Rx/7Sx SAI features</td>
<td></td>
</tr>
<tr>
<td>585</td>
<td>SAI internal input/output signals</td>
<td></td>
</tr>
<tr>
<td>586</td>
<td>SAI input/output pins</td>
<td></td>
</tr>
<tr>
<td>587</td>
<td>External synchronization selection</td>
<td></td>
</tr>
<tr>
<td>588</td>
<td>MCLK_x activation conditions</td>
<td></td>
</tr>
<tr>
<td>589</td>
<td>Clock generator programming examples</td>
<td></td>
</tr>
<tr>
<td>590</td>
<td>TDM settings</td>
<td></td>
</tr>
<tr>
<td>591</td>
<td>TDM frame configuration examples</td>
<td></td>
</tr>
<tr>
<td>592</td>
<td>SOPD pattern</td>
<td></td>
</tr>
<tr>
<td>593</td>
<td>Parity bit calculation</td>
<td></td>
</tr>
<tr>
<td>594</td>
<td>Audio sampling frequency versus symbol rates</td>
<td></td>
</tr>
<tr>
<td>595</td>
<td>SAI interrupt sources</td>
<td></td>
</tr>
<tr>
<td>596</td>
<td>SAI register map and reset values</td>
<td></td>
</tr>
<tr>
<td>597</td>
<td>SPDIFRX internal input/output signals</td>
<td></td>
</tr>
<tr>
<td>598</td>
<td>SPDIFRX pins</td>
<td></td>
</tr>
<tr>
<td>599</td>
<td>Transition sequence for preamble</td>
<td></td>
</tr>
<tr>
<td>600</td>
<td>Minimum spdifrx_ker_ck frequency versus audio sampling rate</td>
<td></td>
</tr>
<tr>
<td>601</td>
<td>Conditions of spdifrx_symb_ck generation</td>
<td></td>
</tr>
<tr>
<td>602</td>
<td>SPDIFRX interface register map and reset values</td>
<td></td>
</tr>
<tr>
<td>603</td>
<td>MDIOS input/output signals connected to package pins or balls</td>
<td></td>
</tr>
<tr>
<td>604</td>
<td>MDIOS internal input/output signals</td>
<td></td>
</tr>
<tr>
<td>605</td>
<td>Interrupt control bits</td>
<td></td>
</tr>
<tr>
<td>606</td>
<td>MDIOS register map and reset values</td>
<td></td>
</tr>
<tr>
<td>607</td>
<td>SDMMC features</td>
<td></td>
</tr>
<tr>
<td>608</td>
<td>SDMMC operation modes SD and SDIO</td>
<td></td>
</tr>
<tr>
<td>609</td>
<td>SDMMC operation modes e+MMC</td>
<td></td>
</tr>
</tbody>
</table>
Table 664. OTG_HS implementation ................................. 3055
Table 661. Minimum duration for soft disconnect ................. 2969
Table 645. Standard message ID filter element ....................... 2858
Table 621. Command path status flags ............................. 2762
Table 620. Specific Commands overview ............................. 2761
Table 619. Long response with CRC token format .................... 2760
Table 618. Short response without CRC token format ............... 2760
Table 617. Short response with CRC token format ................. 2760
Table 616. Command token format ................................ 2759
Table 615. SDMMC Command and data phase selection ............... 2753
Table 614. SDMMC pins ............................................. 2752
Table 613. SDMMC internal input/output signals ...................... 2751

Table 663. OTG_HS speeds supported ............................... 3053
Table 659. Power and clock gating control and status registers .... 2922
Table 658. Data FIFO (DFIFO) access register map ................. 2922
Table 657. Device-mode control and status registers ............... 2920
Table 655. Core global control and status registers (CSRs) ......... 2918
Table 654. Compatibility of STM32 low power modes with the OTG .. 2910
Table 653. OTG_FS input/output signals ............................ 2899
Table 652. OTG_FS input/output pins ............................... 2898
Table 650. OTG_FS speeds supported ................................ 2894
Table 649. FDCAN register map and reset values .................... 2890
Table 648. Extended message ID filter element field description .... 2857
Table 647. Extended message ID filter element ...................... 2856
Table 646. Standard message ID filter element field description ...... 2855
Table 645. Standard message ID filter element ...................... 2854
Table 644. Tx event FIFO element ................................ 2852
Table 643. Tx event FIFO element description ......................... 2851
Table 642. Tx buffer element description ......................... 2850
Table 641. Tx buffer and FIFO element ............................. 2849
Table 640. Rx FIFO element ....................................... 2848
Table 639. Rx FIFO element ....................................... 2847
Table 638. Possible configurations for frame transmission .......... 2846
Table 637. DLC coding in FDCAN ................................ 2845
Table 636. CAN subsystem I/O signals .............................. 2844
Table 635. SDMMC register map ..................................... 2843
Table 634. Response type and SDMMC_RESPxR registers .......... 2842
Table 633. SDMMC interrupts ....................................... 2841
Table 632. CMD12 use cases ....................................... 2839
Table 630. SDIO special operation control .......................... 2789
Table 629. AHB and SDMMC_CK clock frequency relation ............ 2788
Table 628. Receive FIFO status flags ................................ 2787
Table 627. Transmit FIFO status flags .............................. 2786
Table 626. Data FIFO access ....................................... 2785
Table 625. Data path error handling ................................ 2784
Table 624. Data path status flags and clear bits .................. 2783
Table 623. Data token format ...................................... 2782
Table 622. Command path error handling ............................ 2781
Table 621. Command path status flags .............................. 2780
Table 620. Specific Commands overview ............................. 2779
Table 619. Long response with CRC token format .................... 2778
Table 618. Short response without CRC token format ............... 2777
Table 617. Short response with CRC token format .................... 2776
Table 616. Command token format ................................ 2775
Table 615. SDMMC Command and data phase selection ............... 2774
Table 614. SDMMC pins ............................................. 2773
Table 613. SDMMC internal input/output signals ...................... 2772

Table 623. Data token format ...................................... 2771
Table 622. Data path error handling ................................ 2770
Table 621. Command path status flags .............................. 2769
Table 620. Specific Commands overview ............................. 2768
Table 619. Long response with CRC token format .................... 2767
Table 618. Short response without CRC token format ............... 2766
Table 617. Short response with CRC token format .................... 2765
Table 616. Command token format ................................ 2764
Table 615. SDMMC Command and data phase selection ............... 2763
Table 614. SDMMC pins ............................................. 2762
Table 613. SDMMC internal input/output signals ...................... 2761

Table 642. Tx buffer element description ......................... 2854
Table 641. Tx buffer and FIFO element ............................. 2853
Table 640. Rx FIFO element ....................................... 2852
Table 639. Rx FIFO element ....................................... 2851
Table 638. Possible configurations for frame transmission .......... 2850
Table 637. DLC coding in FDCAN ................................ 2849
Table 636. CAN subsystem I/O signals .............................. 2848
Table 635. SDMMC register map ..................................... 2847
Table 634. Response type and SDMMC_RESPxR registers .......... 2846
Table 633. SDMMC interrupts ....................................... 2845
Table 632. CMD12 use cases ....................................... 2844
Table 630. SDIO special operation control .......................... 2843
Table 629. AHB and SDMMC_CK clock frequency relation ............ 2842
Table 628. Receive FIFO status flags ................................ 2841
Table 627. Transmit FIFO status flags .............................. 2840
Table 626. Data FIFO access ....................................... 2839
Table 625. Data path error handling ................................ 2838
Table 624. Data path status flags and clear bits .................. 2837
Table 623. Data token format ...................................... 2836
Table 622. Command path error handling ............................ 2835
Table 621. Command path status flags .............................. 2834
Table 620. Specific Commands overview ............................. 2833
Table 619. Long response with CRC token format .................... 2832
Table 618. Short response without CRC token format ............... 2831
Table 617. Short response with CRC token format .................... 2830
Table 616. Command token format ................................ 2829
Table 615. SDMMC Command and data phase selection ............... 2828
Table 614. SDMMC pins ............................................. 2827
Table 613. SDMMC internal input/output signals ...................... 2826

Table 633. SDMMC interrupts ....................................... 2825
Table 632. CMD12 use cases ....................................... 2824
Table 630. SDIO special operation control .......................... 2823
Table 629. AHB and SDMMC_CK clock frequency relation ............ 2822
Table 628. Receive FIFO status flags ................................ 2821
Table 627. Transmit FIFO status flags .............................. 2820
Table 626. Data FIFO access ....................................... 2819
Table 625. Data path error handling ................................ 2818
Table 624. Data path status flags and clear bits .................. 2817
Table 623. Data token format ...................................... 2816
Table 622. Command path error handling ............................ 2815
Table 621. Command path status flags .............................. 2814
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>665</td>
<td>OTG_HS input/output pins</td>
<td>3056</td>
</tr>
<tr>
<td>666</td>
<td>OTG_HS input/output signals</td>
<td>3057</td>
</tr>
<tr>
<td>667</td>
<td>Compatibility of STM32 low power modes with the OTG</td>
<td>3067</td>
</tr>
<tr>
<td>668</td>
<td>Core global control and status registers (CSRs)</td>
<td>3076</td>
</tr>
<tr>
<td>669</td>
<td>Host-mode control and status registers (CSRs)</td>
<td>3077</td>
</tr>
<tr>
<td>670</td>
<td>Device-mode control and status registers</td>
<td>3078</td>
</tr>
<tr>
<td>671</td>
<td>Data FIFO (DFIFO) access register map</td>
<td>3080</td>
</tr>
<tr>
<td>672</td>
<td>Power and clock gating control and status registers</td>
<td>3081</td>
</tr>
<tr>
<td>673</td>
<td>TRDT values</td>
<td>3087</td>
</tr>
<tr>
<td>674</td>
<td>Minimum duration for soft disconnect</td>
<td>3130</td>
</tr>
<tr>
<td>675</td>
<td>OTG_FS/OTG_HS register map and reset values</td>
<td>3156</td>
</tr>
<tr>
<td>676</td>
<td>UCPD implementation</td>
<td>3223</td>
</tr>
<tr>
<td>677</td>
<td>UCPD software trim data</td>
<td>3223</td>
</tr>
<tr>
<td>678</td>
<td>UCPD signals on pins</td>
<td>3224</td>
</tr>
<tr>
<td>679</td>
<td>UCPD internal signals</td>
<td>3225</td>
</tr>
<tr>
<td>680</td>
<td>4b5b symbol encoding table</td>
<td>3226</td>
</tr>
<tr>
<td>681</td>
<td>Ordered sets</td>
<td>3228</td>
</tr>
<tr>
<td>682</td>
<td>Validation of ordered sets</td>
<td>3228</td>
</tr>
<tr>
<td>683</td>
<td>Data size</td>
<td>3229</td>
</tr>
<tr>
<td>684</td>
<td>Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx</td>
<td>3237</td>
</tr>
<tr>
<td>685</td>
<td>Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)</td>
<td>3239</td>
</tr>
<tr>
<td>686</td>
<td>Effect of low power modes on the UCPD</td>
<td>3241</td>
</tr>
<tr>
<td>687</td>
<td>UCPD interrupt requests</td>
<td>3242</td>
</tr>
<tr>
<td>688</td>
<td>UCPD register map and reset values</td>
<td>3259</td>
</tr>
<tr>
<td>689</td>
<td>Ethernet peripheral pins</td>
<td>3265</td>
</tr>
<tr>
<td>690</td>
<td>Ethernet internal input/output signals</td>
<td>3266</td>
</tr>
<tr>
<td>691</td>
<td>Priority scheme for Tx DMA and Rx DMA</td>
<td>3275</td>
</tr>
<tr>
<td>692</td>
<td>Double VLAN processing features in Tx path</td>
<td>3281</td>
</tr>
<tr>
<td>693</td>
<td>Double VLAN processing in Rx path</td>
<td>3282</td>
</tr>
<tr>
<td>694</td>
<td>VLAN insertion or replacement based on VLTI bit</td>
<td>3283</td>
</tr>
<tr>
<td>695</td>
<td>Destination address filtering</td>
<td>3286</td>
</tr>
<tr>
<td>696</td>
<td>Source address filtering</td>
<td>3287</td>
</tr>
<tr>
<td>697</td>
<td>VLAN match status</td>
<td>3288</td>
</tr>
<tr>
<td>698</td>
<td>Ordinary clock: PTP messages for snapshot</td>
<td>3291</td>
</tr>
<tr>
<td>699</td>
<td>End-to-end transparent clock: PTP messages for snapshot</td>
<td>3292</td>
</tr>
<tr>
<td>700</td>
<td>Peer-to-peer transparent clock: PTP messages for snapshot</td>
<td>3293</td>
</tr>
<tr>
<td>701</td>
<td>Egress and ingress latency for PHY interfaces</td>
<td>3296</td>
</tr>
<tr>
<td>702</td>
<td>Minimum PTP clock frequency example</td>
<td>3297</td>
</tr>
<tr>
<td>703</td>
<td>Message format defined in IEEE 1588-2008</td>
<td>3298</td>
</tr>
<tr>
<td>704</td>
<td>Message format defined in IEEE 1588-2008</td>
<td>3298</td>
</tr>
<tr>
<td>705</td>
<td>IPv6-UDP PTP packet fields required for control and status</td>
<td>3299</td>
</tr>
<tr>
<td>706</td>
<td>Ethernet PTP packet fields required for control and status</td>
<td>3300</td>
</tr>
<tr>
<td>707</td>
<td>Timestamp Snapshot Dependency on ETH_MACTSCR bits</td>
<td>3302</td>
</tr>
<tr>
<td>708</td>
<td>PTP message generation criteria</td>
<td>3308</td>
</tr>
<tr>
<td>709</td>
<td>Common PTP message header fields</td>
<td>3310</td>
</tr>
<tr>
<td>710</td>
<td>MAC Transmit PTP mode and one-step timestamping operation</td>
<td>3313</td>
</tr>
<tr>
<td>711</td>
<td>Transmit checksum offload engine functions for different packet types</td>
<td>3318</td>
</tr>
<tr>
<td>712</td>
<td>Receive checksum offload engine functions for different packet types</td>
<td>3320</td>
</tr>
<tr>
<td>713</td>
<td>TSO: TCP and IP header fields</td>
<td>3324</td>
</tr>
<tr>
<td>714</td>
<td>Pause packet fields</td>
<td>3329</td>
</tr>
<tr>
<td>715</td>
<td>Tx MAC flow control</td>
<td>3330</td>
</tr>
<tr>
<td>716</td>
<td>Rx MAC flow control</td>
<td>3330</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>717.</td>
<td>Size of the maximum receive packet</td>
<td>3333</td>
</tr>
<tr>
<td>718.</td>
<td>MCD clock selection</td>
<td>3336</td>
</tr>
<tr>
<td>719.</td>
<td>MDIO Clause 45 frame structure</td>
<td>3337</td>
</tr>
<tr>
<td>720.</td>
<td>MDIO Clause 22 frame structure</td>
<td>3338</td>
</tr>
<tr>
<td>721.</td>
<td>Remote wake-up packet filter register</td>
<td>3349</td>
</tr>
<tr>
<td>722.</td>
<td>Description of the remote wake-up filter fields</td>
<td>3350</td>
</tr>
<tr>
<td>723.</td>
<td>Remote wake-up packet and PMT interrupt generation</td>
<td>3351</td>
</tr>
<tr>
<td>724.</td>
<td>Transfer complete interrupt behavior</td>
<td>3359</td>
</tr>
<tr>
<td>725.</td>
<td>TDES0 normal descriptor (read format)</td>
<td>3380</td>
</tr>
<tr>
<td>726.</td>
<td>TDES1 normal descriptor (read format)</td>
<td>3381</td>
</tr>
<tr>
<td>727.</td>
<td>TDES2 normal descriptor (read format)</td>
<td>3381</td>
</tr>
<tr>
<td>728.</td>
<td>TDES3 normal descriptor (read format)</td>
<td>3382</td>
</tr>
<tr>
<td>729.</td>
<td>TDES0 normal descriptor (write-back format)</td>
<td>3385</td>
</tr>
<tr>
<td>730.</td>
<td>TDES1 normal descriptor (write-back format)</td>
<td>3385</td>
</tr>
<tr>
<td>731.</td>
<td>TDES2 normal descriptor (write-back format)</td>
<td>3386</td>
</tr>
<tr>
<td>732.</td>
<td>TDES3 normal descriptor (write-back format)</td>
<td>3386</td>
</tr>
<tr>
<td>733.</td>
<td>TDES0 context descriptor</td>
<td>3389</td>
</tr>
<tr>
<td>734.</td>
<td>TDES1 context descriptor</td>
<td>3390</td>
</tr>
<tr>
<td>735.</td>
<td>TDES2 context descriptor</td>
<td>3390</td>
</tr>
<tr>
<td>736.</td>
<td>TDES3 context descriptor</td>
<td>3390</td>
</tr>
<tr>
<td>737.</td>
<td>RDES0 normal descriptor (read format)</td>
<td>3394</td>
</tr>
<tr>
<td>738.</td>
<td>RDES1 normal descriptor (read format)</td>
<td>3394</td>
</tr>
<tr>
<td>739.</td>
<td>RDES2 normal descriptor (read format)</td>
<td>3394</td>
</tr>
<tr>
<td>740.</td>
<td>RDES3 normal descriptor (read format)</td>
<td>3394</td>
</tr>
<tr>
<td>741.</td>
<td>RDES0 normal descriptor (write-back format)</td>
<td>3396</td>
</tr>
<tr>
<td>742.</td>
<td>RDES1 normal descriptor (write-back format)</td>
<td>3396</td>
</tr>
<tr>
<td>743.</td>
<td>RDES2 normal descriptor (write-back format)</td>
<td>3398</td>
</tr>
<tr>
<td>744.</td>
<td>RDES3 normal descriptor (write-back format)</td>
<td>3400</td>
</tr>
<tr>
<td>745.</td>
<td>RDES0 context descriptor</td>
<td>3403</td>
</tr>
<tr>
<td>746.</td>
<td>RDES1 context descriptor</td>
<td>3404</td>
</tr>
<tr>
<td>747.</td>
<td>RDES2 context descriptor</td>
<td>3404</td>
</tr>
<tr>
<td>748.</td>
<td>RDES3 context descriptor</td>
<td>3404</td>
</tr>
<tr>
<td>749.</td>
<td>ETH_DMA common register map and reset values</td>
<td>3426</td>
</tr>
<tr>
<td>750.</td>
<td>ETH_DMA_CH register map and reset values</td>
<td>3426</td>
</tr>
<tr>
<td>751.</td>
<td>ETH_MTL register map and reset values</td>
<td>3439</td>
</tr>
<tr>
<td>752.</td>
<td>Giant Packet Status based on S2KP and JE Bits</td>
<td>3445</td>
</tr>
<tr>
<td>753.</td>
<td>Packet Length based on the CST and ACS bits</td>
<td>3445</td>
</tr>
<tr>
<td>754.</td>
<td>Ethernet MAC register map and reset values</td>
<td>3526</td>
</tr>
<tr>
<td>755.</td>
<td>HDMI pin</td>
<td>3538</td>
</tr>
<tr>
<td>756.</td>
<td>HDMI-CEC internal input/output signals</td>
<td>3538</td>
</tr>
<tr>
<td>757.</td>
<td>Error handling timing parameters</td>
<td>3544</td>
</tr>
<tr>
<td>758.</td>
<td>TXERR timing parameters</td>
<td>3545</td>
</tr>
<tr>
<td>759.</td>
<td>HDMI-CEC interrupts</td>
<td>3546</td>
</tr>
<tr>
<td>760.</td>
<td>HDMI-CEC register map and reset values</td>
<td>3554</td>
</tr>
<tr>
<td>761.</td>
<td>Authentication signal states</td>
<td>3559</td>
</tr>
<tr>
<td>762.</td>
<td>Authentication signal states</td>
<td>3559</td>
</tr>
<tr>
<td>763.</td>
<td>Packet request</td>
<td>3562</td>
</tr>
<tr>
<td>764.</td>
<td>ACK response</td>
<td>3562</td>
</tr>
<tr>
<td>765.</td>
<td>Data transfer</td>
<td>3563</td>
</tr>
<tr>
<td>766.</td>
<td>JTAG-DP data registers</td>
<td>3565</td>
</tr>
<tr>
<td>767.</td>
<td>Debug port registers</td>
<td>3567</td>
</tr>
<tr>
<td>768.</td>
<td>Debug port register map and reset values</td>
<td>3576</td>
</tr>
</tbody>
</table>
Table 769. MEM-AP registers ................................................................. 3578
Table 770. Access port register map and reset values ................................. 3586
Table 771. System ROM table 1 ............................................................. 3588
Table 772. System ROM table 2 ............................................................. 3588
Table 773. System ROM table 1 register map and reset values ...................... 3595
Table 774. System ROM table 2 register map and reset values ...................... 3596
Table 775. System CTI inputs ............................................................... 3598
Table 776. System CTI outputs ............................................................. 3598
Table 777. Cortex-M7 CTI inputs .......................................................... 3598
Table 778. Cortex-M7 CTI outputs ........................................................ 3599
Table 779. CTI register map and reset values ........................................... 3614
Table 780. CSTF register map and reset values ........................................ 3626
Table 781. ETF register map and reset values .......................................... 3647
Table 782. TPIU register map and reset values ......................................... 3666
Table 783. SWO register map and reset values ...................................... 3678
Table 784. DBGMCU register map and reset values ................................... 3695
Table 785. Cortex-M7 CPU ROM table .................................................. 3698
Table 786. Cortex-M7 PPB ROM table .................................................... 3699
Table 787. Cortex-M7 CPU ROM table register map and reset values ............. 3705
Table 788. Cortex-M7 PPB ROM table register map and reset values ............. 3710
Table 789. Cortex-M7 DWT register map and reset values .......................... 3721
Table 790. Cortex-M7 ITM register map and reset values ........................... 3730
Table 791. Cortex-M7 FPB register map and reset values ........................... 3737
Table 792. Cortex-M7 ETM register map and reset values ........................... 3765
Table 793. Document revision history ................................................... 3775
# List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>System architecture</td>
<td>123</td>
</tr>
<tr>
<td>Figure 2</td>
<td>AXI interconnect</td>
<td>128</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Memory map</td>
<td>150</td>
</tr>
<tr>
<td>Figure 4</td>
<td>RAM ECC controller implementation schematic</td>
<td>165</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Connection between RAM ECC controller and RAMECC monitoring unit</td>
<td>166</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Secure boot process</td>
<td>181</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Key management principle</td>
<td>187</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Product life-cycle security</td>
<td>190</td>
</tr>
<tr>
<td>Figure 9</td>
<td>FLASH block diagram</td>
<td>200</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Detailed FLASH architecture</td>
<td>202</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Embedded flash memory organization</td>
<td>203</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Embedded flash memory usage (STM32H7R)</td>
<td>205</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Embedded flash memory usage (STM32H7S)</td>
<td>206</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Flash decommissioning options</td>
<td>207</td>
</tr>
<tr>
<td>Figure 15</td>
<td>FLASH read pipeline architecture</td>
<td>209</td>
</tr>
<tr>
<td>Figure 16</td>
<td>FLASH write pipeline architecture</td>
<td>213</td>
</tr>
<tr>
<td>Figure 17</td>
<td>FLASH stateful initialization</td>
<td>220</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Life cycle management scheme</td>
<td>232</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Power control block diagram</td>
<td>279</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Power supply overview</td>
<td>283</td>
</tr>
<tr>
<td>Figure 21</td>
<td>System supply configurations</td>
<td>284</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Device startup with VCORE supplied directly from SMPS step-down converter</td>
<td>287</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Device startup with VCORE supplied in Bypass mode</td>
<td>288</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Backup domain</td>
<td>293</td>
</tr>
<tr>
<td>Figure 25</td>
<td>USB supply configurations</td>
<td>295</td>
</tr>
<tr>
<td>Figure 26</td>
<td>Power-on reset/power-down reset waveform</td>
<td>296</td>
</tr>
<tr>
<td>Figure 27</td>
<td>BOR thresholds</td>
<td>297</td>
</tr>
<tr>
<td>Figure 28</td>
<td>PVD thresholds</td>
<td>298</td>
</tr>
<tr>
<td>Figure 29</td>
<td>AVD thresholds</td>
<td>299</td>
</tr>
<tr>
<td>Figure 30</td>
<td>VBAT thresholds</td>
<td>300</td>
</tr>
<tr>
<td>Figure 31</td>
<td>Temperature thresholds</td>
<td>301</td>
</tr>
<tr>
<td>Figure 32</td>
<td>VCORE overvoltage protection</td>
<td>302</td>
</tr>
<tr>
<td>Figure 33</td>
<td>$V_{CORE}$ voltage scaling versus system power modes</td>
<td>305</td>
</tr>
<tr>
<td>Figure 34</td>
<td>Dynamic voltage scaling in Run mode</td>
<td>306</td>
</tr>
<tr>
<td>Figure 35</td>
<td>Dynamic voltage scaling behavior in Stop mode</td>
<td>307</td>
</tr>
<tr>
<td>Figure 36</td>
<td>Dynamic voltage scaling from Standby mode</td>
<td>308</td>
</tr>
<tr>
<td>Figure 37</td>
<td>RCC block diagram</td>
<td>341</td>
</tr>
<tr>
<td>Figure 38</td>
<td>Simplified reset circuit</td>
<td>345</td>
</tr>
<tr>
<td>Figure 39</td>
<td>Boot sequences versus system states</td>
<td>351</td>
</tr>
<tr>
<td>Figure 40</td>
<td>Top-level clock tree</td>
<td>353</td>
</tr>
<tr>
<td>Figure 41</td>
<td>HSE/LSE clock source</td>
<td>355</td>
</tr>
<tr>
<td>Figure 42</td>
<td>HSE clock generation</td>
<td>355</td>
</tr>
<tr>
<td>Figure 43</td>
<td>LSE clock generation</td>
<td>357</td>
</tr>
<tr>
<td>Figure 44</td>
<td>HSI calibration flow</td>
<td>361</td>
</tr>
<tr>
<td>Figure 45</td>
<td>CSI calibration flow</td>
<td>362</td>
</tr>
<tr>
<td>Figure 46</td>
<td>HSI48 calibration flow</td>
<td>363</td>
</tr>
</tbody>
</table>
Figure 47. LSE and HSE CSS function .............................................................. 364
Figure 48. PLL block diagram ........................................................................ 370
Figure 49. Spread-spectrum modulation .......................................................... 374
Figure 50. Triangular waveform generator ......................................................... 375
Figure 51. Core and bus clock generation ......................................................... 377
Figure 52. Key signals controlling low-power modes ........................................ 379
Figure 53. Clock distribution for SAIx, ADx and SPDIFRx ............................... 390
Figure 54. Clock distribution for SPIx and SPI/I2S ........................................... 391
Figure 55. Clock distribution for I2C[3:0] and I2C1/3C1 ................................. 392
Figure 56. Clock distribution for UARTx, LPUART1 and USARTx ............... 393
Figure 57. Clock distribution for FDCAN ......................................................... 394
Figure 58. Clock distribution for GPU2D ......................................................... 395
Figure 59. Clock distribution for LTDC and DCMIPP ...................................... 395
Figure 60. Clock distribution for PSSI .............................................................. 396
Figure 61. Clock distribution for FMC, XSPIs, and MCEs ............................... 388
Figure 62. Clock distribution for SDMMC[2:1] and DB_SDMMC[2:1] ............. 399
Figure 63. Clock distribution for USB and UCPD .......................................... 401
Figure 64. Clock distribution for ETH1 ............................................................ 402
Figure 65. ETH clock configuration ............................................................... 403
Figure 66. Clock distribution for ADCs ........................................................... 404
Figure 67. Clock distribution for SAEx and RNG ............................................. 404
Figure 68. Clock distribution for HDMI-CEC .................................................. 405
Figure 69. Clock distribution for TIMs ............................................................ 405
Figure 70. Clock distribution for LPTIMs ........................................................ 407
Figure 71. Clock distribution for RTC ............................................................. 408
Figure 72. Clock distribution for WWGD and IWGD ....................................... 409
Figure 73. Clock distribution for DBG and trace .......................................... 409
Figure 74. Kernel clock switching ................................................................. 411
Figure 75. Peripheral kernel clock enable logic details .................................... 412
Figure 76. SBS block diagram ...................................................................... 540
Figure 77. SBS boot control ........................................................................... 541
Figure 78. Compensation cell management .................................................... 544
Figure 79. Compensation cell use ................................................................... 544
Figure 80. CRS block diagram ...................................................................... 561
Figure 81. CRS counter behavior ................................................................... 563
Figure 82. Structure of three-volt or five-volt tolerant GPIO (TT or FT) ........... 572
Figure 83. Input floating/pull-up/pull-down configurations ............................. 576
Figure 84. Output configuration ..................................................................... 577
Figure 85. Alternate function configuration .................................................... 578
Figure 86. High-impedance analog configuration .......................................... 578
Figure 87. GPDMA block diagram ................................................................. 608
Figure 88. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) ................................................................. 609
Figure 89. GPDMA channel suspend and resume sequence .......................... 610
Figure 90. GPDMA channel abort and restart sequence ............................... 611
Figure 91. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x .......................................................... 613
Figure 92. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x .......................................................... 614
Figure 93. GPDMA dynamic linked-list data structure of a linear addressing channel x .......................................................... 615
Figure 94. GPDMA dynamic linked-list data structure of a 2D addressing channel x .......................................................... 615
Figure 95. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) ........................................ 617
Figure 96. Inserting a LLIn with an auxiliary GPDMA channel y .................................................. 619
Figure 97. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) ........................................ 621
Figure 98. Building LLIn+1: GPDMA dynamic linked-lists in link step mode .............................. 622
Figure 99. Replace with a new LLIn' in register file in link step mode ........................................ 623
Figure 100. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) .......... 624
Figure 101. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) ......... 625
Figure 102. GPDMA channel execution and linked-list programming ........................................ 627
Figure 103. Programmed 2D addressing ......................................................................................... 630
Figure 104. GPDMA arbitration policy ......................................................................................... 637
Figure 105. Trigger hit, memorization and overrun waveform ..................................................... 640
Figure 106. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel ........................................ 641
Figure 107. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel ........................................ 642
Figure 108. HPDMA block diagram .......................................................................................... 681
Figure 109. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0) ......... 682
Figure 110. HPDMA channel suspend and resume sequence ..................................................... 683
Figure 111. HPDMA channel abort and restart sequence .......................................................... 684
Figure 112. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x .......... 685
Figure 113. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x ............... 686
Figure 114. HPDMA dynamic linked-list data structure of a linear addressing channel x ............ 687
Figure 115. HPDMA dynamic linked-list data structure of a 2D addressing channel x ............... 687
Figure 116. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0) ................................. 690
Figure 117. Inserting a LLIn with an auxiliary HPDMA channel y .................................................. 692
Figure 118. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1) ........................................ 694
Figure 119. Building LLIn+1: HPDMA dynamic linked-lists in link step mode ......................... 695
Figure 120. Replace with a new LLIn' in register file in link step mode ....................................... 696
Figure 121. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) .......... 697
Figure 122. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) ......... 698
Figure 123. HPDMA channel execution and linked-list programming ........................................ 700
Figure 124. Programmed 2D addressing ..................................................................................... 703
Figure 125. HPDMA arbitration policy ....................................................................................... 718
Figure 126. Trigger hit, memorization and overrun waveform ..................................................... 722
Figure 127. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel ........................................ 723
Figure 128. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel ........................................ 724
Figure 129. GFXMMU block diagram .......................................................................................... 758
Figure 130. Virtual buffer ........................................................................................................ 760
Figure 131. Virtual buffer and physical buffer memory map ....................................................... 761
Figure 132. MMU block diagram ................................................................................................... 762
Figure 133. Block validation/comparator implementation .............................................................. 764
Figure 134. DMA2D block diagram .............................................................................................. 775
Figure 135. Intel 8080 16-bit mode (RGB565) .............................................................................. 782
Figure 136. Intel 8080 18/24-bit mode (RGB888) ........................................................................... 782
Figure 137. GPU2D block diagram .............................................................................................. 809
Figure 138. ICACHE block diagram ............................................................................................ 813
List of figures

Figure 139. ICACHE TAG and data memories functional view .................................................. 815
Figure 140. GFXTIM block diagram .......................................................................................... 825
Figure 141. Clock generator ..................................................................................................... 827
Figure 142. Waveforms in standalone ...................................................................................... 829
Figure 143. Active counters and signals in standalone ............................................................. 829
Figure 144. Waveforms with external HSYNC and VSYNC. .................................................. 829
Figure 145. Waveforms with external HSYNC only ............................................................... 830
Figure 146. Active counters and signals with external HSYNC only ........................................ 830
Figure 147. Waveforms with external VSYNC only ............................................................... 830
Figure 148. Active counters with external VSYNC only ....................................................... 831
Figure 149. Prescaling when external VSYNC only .............................................................. 831
Figure 150. Waveforms with external CSYNC only ............................................................... 831
Figure 151. Active counters and signals with external CSYNC only ....................................... 832
Figure 152. Prescaling when external CSYNC only .............................................................. 832
Figure 153. Tearing-effect configurations ............................................................................... 834
Figure 154. Watchdog timer ................................................................................................... 835
Figure 155. EXTI block diagram ............................................................................................ 869
Figure 156. Configurable event triggering logic CPU wakeup ................................................ 871
Figure 157. Direct event triggering logic CPU wakeup ........................................................... 872
Figure 158. CRC calculation unit block diagram ..................................................................... 889
Figure 159. CORDIC convergence for trigonometric functions .............................................. 903
Figure 160. CORDIC convergence for hyperbolic functions .................................................. 904
Figure 161. CORDIC convergence for square root .................................................................. 905
Figure 162. FMC block diagram ............................................................................................ 915
Figure 163. FMC memory banks (default mapping) ............................................................... 918
Figure 164. Mode 1 read access waveforms ............................................................................ 929
Figure 165. Mode 1 write access waveforms .......................................................................... 930
Figure 166. Mode A read access waveforms ........................................................................... 932
Figure 167. Mode A write access waveforms .......................................................................... 933
Figure 168. Mode 2 and mode B read access waveforms ....................................................... 935
Figure 169. Mode 2 write access waveforms .......................................................................... 935
Figure 170. Mode B write access waveforms .......................................................................... 936
Figure 171. Mode C read access waveforms .......................................................................... 938
Figure 172. Mode C write access waveforms .......................................................................... 938
Figure 173. Mode D read access waveforms .......................................................................... 941
Figure 174. Mode D write access waveforms .......................................................................... 941
Figure 175. Muxed read access waveforms ............................................................................ 944
Figure 176. Muxed write access waveforms .......................................................................... 944
Figure 177. Asynchronous wait during a read access waveforms ........................................... 947
Figure 178. Asynchronous wait during a write access waveforms .......................................... 947
Figure 179. Wait configuration waveforms .............................................................................. 950
Figure 180. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) .......... 950
Figure 181. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) .................. 952
Figure 182. NAND flash controller waveforms for common memory access ....................... 966
Figure 183. Access to non ‘CE don’t care’ NAND-flash ......................................................... 967
Figure 184. Burst write SDRAM access waveforms ............................................................... 977
Figure 185. Burst read SDRAM access ................................................................................... 978
Figure 186. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0) .......... 979
Figure 187. Read access crossing row boundary ..................................................................... 981
Figure 188. Write access crossing row boundary .................................................................... 981
Figure 189. Self-refresh mode ............................................................................................... 984
Figure 190. Power-down mode .............................................................................................. 985
Figure 191. XSPI block diagram for 16-bit configuration \( ^{(1)} \) ......................................................... 997
Figure 192. XSPI block diagram for dual-octal configuration \( ^{(1)} \) ......................................................... 998
Figure 193. XSPI block diagram for octal configuration ........................................................ 999
Figure 194. XSPI block diagram in quad configuration .......................................................... 1000
Figure 195. XSPI block diagram for dual-quad configuration ................................................. 1001
Figure 196. SDR read command in 16-bit configuration ....................................................... 1003
Figure 197. DTR read in octal-SPI mode with DQS (Macronix mode) example ......................... 1006
Figure 198. SDR write command in octal-SPI mode example ............................................. 1008
Figure 199. DTR write in octal-SPI mode (Macronix mode) example .................................. 1009
Figure 200. Example of HyperBus read operation (8-bit data mode) ..................................... 1011
Figure 201. HyperBus write operation with initial latency (8-bit data mode) .......................... 1013
Figure 202. HyperBus read operation with additional latency (8-bit data mode) .................. 1013
Figure 203. HyperBus write operation with additional latency (8-bit data mode) .............. 1014
Figure 204. HyperBus write operation with no latency (register write) ............................... 1014
Figure 205. HyperBus read operation page crossing with latency (8-bit data mode) ............. 1015
Figure 206. HyperBus write operation with initial latency (16-bit mode) ............................. 1016
Figure 207. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access ............... 1022
Figure 208. OctaRAM read operation with reverse data ordering D1/D0 ............................. 1023
Figure 209. NCS when CKMODE = 0 (T = CLK period) ...................................................... 1029
Figure 210. NCS when CKMODE = 1 in SDR mode (T = CLK period) ............................... 1030
Figure 211. NCS when CKMODE = 1 in DTR mode (T = CLK period) ............................. 1030
Figure 212. NCS when CKMODE =1 with an abort (T = CLK period). ............................... 1030
Figure 213. Example of software-controller dual-memory support ..................................... 1031
Figure 214. XSPI block diagram ......................................................................................... 1063
Figure 215. XSPI octal direct mode .................................................................................... 1065
Figure 216. XSPI direct 16-bit mode .................................................................................. 1066
Figure 217. XSPI dual-octal mode ..................................................................................... 1066
Figure 218. XSPI swapped (octal) mode ............................................................................. 1067
Figure 219. XSPI multiplexed mode to Port 1 ..................................................................... 1068
Figure 220. XSPI multiplexed (octal and dual-octal) mode to Port 2 ................................. 1068
Figure 221. XSPI1 and XSPI2 drive a single external memory (octal mode) ....................... 1069
Figure 222. Single XSPI driving two external memories ..................................................... 1070
Figure 223. DLYB block diagram ....................................................................................... 1073
Figure 224. ADC block diagram ....................................................................................... 1081
Figure 225. ADC clock scheme ........................................................................................ 1086
Figure 226. ADC1 connectivity ......................................................................................... 1087
Figure 227. ADC2 connectivity ........................................................................................ 1088
Figure 228. ADC calibration .............................................................................................. 1091
Figure 229. Updating the ADC calibration factor .................................................................. 1092
Figure 230. Mixing single-ended and differential channels .................................................. 1093
Figure 231. Enabling / disabling the ADC .......................................................................... 1094
Figure 232. Bulb mode timing diagram .............................................................................. 1097
Figure 233. Analog-to-digital conversion time .................................................................... 1100
Figure 234. Stopping ongoing regular conversions .............................................................. 1101
Figure 235. Stopping ongoing regular and injected conversions .......................................... 1102
Figure 236. Triggers shared between ADC master and slave ............................................... 1103
Figure 237. Injected conversion latency ............................................................................... 1105
Figure 238. Example of JSQR queue of context (sequence change) .................................... 1108
Figure 239. Example of JSQR queue of context (trigger change) ....................................... 1108
Figure 240. Example of JSQR queue of context with overflow before conversion .......... 1109
Figure 241. Example of JSQR queue of context with overflow during conversion ............. 1109
Figure 242. Example of JSQR queue of context with empty queue (case JQM = 0) ............ 1110
List of figures

Figure 243. Example of JSQR queue of context with empty queue (JQM = 1) .................. 1111
Figure 244. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion. .................. 1111
Figure 245. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs .................. 1112
Figure 246. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion .................. 1112
Figure 247. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1) ................ 1113
Figure 248. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0) .................. 1113
Figure 249. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 1) .................. 1114
Figure 250. Single conversions of a sequence, software trigger .......................... 1116
Figure 251. Continuous conversion of a sequence, software trigger .................. 1116
Figure 252. Single conversions of a sequence, hardware trigger .................. 1117
Figure 253. Continuous conversions of a sequence, hardware trigger .................. 1117
Figure 254. Right alignment (offset disabled, unsigned value) .................. 1119
Figure 255. Right alignment (offset enabled, signed value) .................. 1120
Figure 256. Left alignment (offset disabled, unsigned value) .................. 1120
Figure 257. Left alignment (offset enabled, signed value) .................. 1121
Figure 258. Example of overrun (OVRMOD = 0) ........................................... 1122
Figure 259. Example of overrun (OVRMOD = 1) ........................................... 1123
Figure 260. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) .................. 1127
Figure 261. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) .................. 1128
Figure 262. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 0) .................. 1128
Figure 263. AUTODLY = 1, regular continuous conversions interrupted by injected conversions ............................. 1129
Figure 264. AUTODLY = 1 in auto-injected mode (JAUTO = 1) .......................... 1129
Figure 265. Analog watchdog guarded area ........................................... 1130
Figure 266. ADCy_AWDx_OUT signal generation (on all regular channels) .................. 1132
Figure 267. ADCy_AWDx_OUT signal generation (AWDx flag not cleared by software) .................. 1133
Figure 268. ADCy_AWDx_OUT signal generation (on a single regular channel) .................. 1133
Figure 269. ADCy_AWDx_OUT signal generation (on all injected channels) .................. 1133
Figure 270. 20-bit to 16-bit result truncation ........................................... 1135
Figure 271. Numerical example with 5-bit shift and rounding ................................ 1135
Figure 272. Triggered regular oversampling mode (TROVS bit = 1) .................. 1137
Figure 273. Regular oversampling modes (4x ratio) ........................................ 1138
Figure 274. Regular and injected oversampling modes used simultaneously .................. 1139
Figure 275. Triggered regular oversampling with injection ................................ 1139
Figure 276. Oversampling in auto-injected mode ........................................... 1140
Figure 277. Dual ADC block diagram(1) ........................................... 1142
Figure 278. Injected simultaneous mode on 4 channels: dual ADC mode .................. 1143
Figure 279. Regular simultaneous mode on 16 channels: dual ADC mode .................. 1145
Figure 280. Interleaved mode on one channel in continuous conversion mode: dual ADC mode .................. 1146
Figure 281. Interleaved mode on 1 channel in single conversion mode: dual ADC mode .................. 1147
Figure 282. Interleaved conversion with injection ........................................... 1147
Figure 283. Alternate trigger: injected group of each ADC ................................ 1148
Figure 284. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode .................. 1149
Figure 285. Alternate + regular simultaneous ........................................... 1150
Figure 286. Case of trigger occurring during injected conversion .................. 1150
Figure 287. Interleaved single channel CH0 with injected sequence CH11, CH12 .................. 1151
Figure 288. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
Figure 289. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 1: Master interrupted first .................................................. 1151
- case 2: Slave interrupted first .................................................... 1151
Figure 290. DMA Requests in regular simultaneous mode when MDMA = 0b00 ........ 1152
Figure 291. DMA requests in regular simultaneous mode when MDMA = 0b10 .......... 1153
Figure 292. DMA requests in interleaved mode when MDMA = 0b10 ................. 1153
Figure 293. Temperature sensor channel block diagram .................................. 1155
Figure 294. VBAT channel block diagram ............................................. 1157
Figure 295. VREFINT channel block diagram ........................................ 1157
Figure 296. Temperature sensor functional block diagram ............................. 1204
Figure 297. Method for low REF_CLK frequencies ...................................... 1206
Figure 298. Method for high REF_CLK frequencies .................................... 1206
Figure 299. Temperature sensor sequence ............................................. 1209
Figure 300. VREFBUF block diagram .................................................. 1220
Figure 301. ADF block diagram .......................................................... 1226
Figure 302. SITF overview ................................................................. 1228
Figure 303. SPI timing example ......................................................... 1229
Figure 304. Manchester timing example (SITFMOD = 11) ............................. 1230
Figure 305. CKGEN overview ............................................................. 1233
Figure 306. BSMX overview ............................................................... 1235
Figure 307. DFLT overview ............................................................... 1236
Figure 308. Programmable delay .......................................................... 1237
Figure 309. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16 1238
Figure 310. Reshape filter frequency response normalized (FRS / 2 = 1) .......... 1244
Figure 311. Trigger logic for DFLT and CKGEN ....................................... 1246
Figure 312. Asynchronous continuous mode (ACQMOD[2:0] = 0) ..................... 1247
Figure 313. Asynchronous single-shot mode (ACQMOD[2:0] = 001) ................. 1248
Figure 314. Synchronous continuous mode (ACQMOD[2:0] = 100) ................. 1249
Figure 315. Synchronous single-shot mode (ACQMOD[2:0] = 011) ................. 1250
Figure 316. Window continuous mode (ACQMOD[2:0] = 100) ........................ 1251
Figure 317. Discard function example .................................................. 1253
Figure 318. Start sequence with DFLTEN, in continuous mode, audio configuration 1253
Figure 319. SAD block diagram .......................................................... 1254
Figure 320. SAD flow diagram ............................................................. 1256
Figure 321. SAD timing diagram example ............................................. 1262
Figure 322. ADF_DFLTxDR data format ............................................. 1262
Figure 323. Data re-synchronization ................................................... 1263
Figure 324. Example of overflow and transfer to memory ................................ 1264
Figure 325. ADF interrupt interface ................................................... 1267
Figure 326. Sensor connection examples ............................................. 1272
Figure 327. Global frequency response .............................................. 1272
Figure 328. Detailed frequency response ............................................ 1273
Figure 329. Simplified DFLT view with gain information ............................. 1275
Figure 330. SAD example working with SADMOD = 01 ............................. 1278
Figure 331. SAD example working with SADMOD = 1x ............................. 1281
Figure 332. DCMIPP overview .......................................................... 1300
Figure 333. DCMIPP block diagram ................................................... 1303
Figure 334. Snapshot (CPTMODE = 1) and Continuous (CPTMODE = 0) capture modes 1313
Figure 335. Pipe0 (dump) architecture overview ..................................... 1316
Figure 336. PSSI block diagram ........................................................ 1362
Figure 337. Top-level block diagram .................................................. 1362
Figure 338. Data enable in receive mode waveform diagram (CKPOL=0) ........... 1366
List of figures

Figure 339.  Data enable waveform diagram in transmit mode (CKPOL=0) .......................... 1366
Figure 340.  Ready in receive mode waveform diagram (CKPOL=0) ............................... 1367
Figure 341.  Bidirectional PSSI_DE/PSSI_RDY waveform .................................................. 1368
Figure 342.  Bidirectional PSSI_DE/PSSI_RDY connection diagram ................................. 1368
Figure 343.  LTDC block diagram ......................................................................................... 1376
Figure 344.  LTDC synchronous timings ............................................................................. 1379
Figure 345.  Layer window programmable parameters ....................................................... 1382
Figure 346.  Blending two layers with background .............................................................. 1384
Figure 347.  Interrupt events ............................................................................................... 1386
Figure 348.  JPEG codec block diagram ........................................................................... 1407
Figure 349.  RNG block diagram ....................................................................................... 1429
Figure 350.  NIST SP800-90B entropy source model ....................................................... 1430
Figure 351.  RNG initialization overview ........................................................................... 1433
Figure 352.  SAES block diagram ..................................................................................... 1448
Figure 353.  Encryption/ decryption typical usage ............................................................... 1450
Figure 354.  Typical operation with authentication .............................................................. 1452
Figure 355.  Example of suspend mode management ........................................................ 1453
Figure 356.  ECB encryption .............................................................................................. 1454
Figure 357.  ECB decryption ............................................................................................... 1454
Figure 358.  CBC encryption ............................................................................................. 1455
Figure 359.  CBC decryption ............................................................................................. 1455
Figure 360.  Message construction in CTR mode .............................................................. 1458
Figure 361.  CTR encryption ............................................................................................. 1459
Figure 362.  Message construction in GCM ....................................................................... 1460
Figure 363.  GCM authenticated encryption .................................................................... 1462
Figure 364.  Message construction in GMAC mode ........................................................... 1465
Figure 365.  GMAC authentication mode .......................................................................... 1465
Figure 366.  Message construction in CCM mode ............................................................ 1466
Figure 367.  CCM mode authenticated encryption ............................................................ 1468
Figure 368.  Operation with wrapped keys for SAES in ECB and CBC modes .................. 1472
Figure 369.  Operation with wrapped keys for SAES in CTR mode ................................... 1474
Figure 370.  Usage of Shared-key mode ........................................................................... 1475
Figure 371.  128-bit block construction according to the data type ...................................... 1478
Figure 372.  CRYP block diagram .................................................................................... 1502
Figure 373.  Encryption/ decryption typical usage .............................................................. 1504
Figure 374.  Typical operation with authentication ............................................................ 1507
Figure 375.  Example of suspend mode management ........................................................ 1508
Figure 376.  ECB encryption ............................................................................................. 1509
Figure 377.  ECB decryption ............................................................................................. 1509
Figure 378.  CBC encryption ............................................................................................. 1510
Figure 379.  CBC decryption ............................................................................................. 1510
Figure 380.  Message construction in CTR mode .............................................................. 1513
Figure 381.  CTR encryption ............................................................................................. 1514
Figure 382.  Message construction in GCM ....................................................................... 1516
Figure 383.  GCM authenticated encryption .................................................................... 1517
Figure 384.  Message construction in GMAC mode ........................................................... 1521
Figure 385.  GMAC authentication mode .......................................................................... 1521
Figure 386.  Message construction in CCM mode ............................................................ 1522
Figure 387.  CCM mode authenticated encryption ............................................................ 1524
Figure 388.  128-bit block construction according to the data type ...................................... 1530
Figure 389.  HASH block diagram ................................................................................... 1551
Figure 390.  Message data swapping feature .................................................................... 1553
Figure 391. HASH suspend/resume mechanism ................................................................. 1559
Figure 392. MCE block diagram ..................................................................................... 1574
Figure 393. MCE region programming ........................................................................... 1575
Figure 394. MCE implementation of block ciphers ......................................................... 1577
Figure 395. MCE implementation of stream cipher ....................................................... 1578
Figure 396. PKA block diagram ....................................................................................... 1597
Figure 397. Advanced-control timer block diagram ...................................................... 1631
Figure 398. Counter timing diagram with prescaler division change from 1 to 2 .......... 1637
Figure 399. Counter timing diagram with prescaler division change from 1 to 4 ......... 1637
Figure 400. Counter timing diagram, internal clock divided by 1 ................................. 1639
Figure 401. Counter timing diagram, internal clock divided by 2 ................................. 1639
Figure 402. Counter timing diagram, internal clock divided by 4 ................................. 1640
Figure 403. Counter timing diagram, internal clock divided by N ............................... 1640
Figure 404. Counter timing diagram, update event when ARPE = 0 (TIMx.ARR not preloaded) ................................................................. 1641
Figure 405. Counter timing diagram, update event when ARPE = 1 (TIMx.ARR preloaded) ................................................................. 1642
Figure 406. Counter timing diagram, internal clock divided by 1 ................................. 1643
Figure 407. Counter timing diagram, internal clock divided by 2 ................................. 1644
Figure 408. Counter timing diagram, internal clock divided by 4 ................................ 1644
Figure 409. Counter timing diagram, internal clock divided by N ............................... 1645
Figure 410. Counter timing diagram, update event when repetition counter is not used 1645
Figure 411. Counter timing diagram, internal clock divided by 1, TIMx.ARR = 0x6 ... 1647
Figure 412. Counter timing diagram, internal clock divided by 2 ................................. 1647
Figure 413. Counter timing diagram, internal clock divided by 4, TIMx.ARR = 0x36 ... 1648
Figure 414. Counter timing diagram, internal clock divided by N ............................... 1648
Figure 415. Counter timing diagram, update event with ARPE = 1 (counter underflow) 1649
Figure 416. Counter timing diagram, Update event with ARPE = 1 (counter overflow) ... 1650
Figure 417. Update rate examples depending on mode and TIMx.RCR register settings 1651
Figure 418. Control circuit in normal mode, internal clock divided by 1 ....................... 1653
Figure 419. tim.til2 external clock connection example .............................................. 1653
Figure 420. Control circuit in external clock mode 1 ................................................... 1654
Figure 421. External trigger input block ........................................................................ 1655
Figure 422. Control circuit in external clock mode 2 ................................................... 1656
Figure 423. Capture/compare channel (example: channel 1 input stage) ....................... 1656
Figure 424. Capture/compare channel 1 main circuit ..................................................... 1657
Figure 425. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) ... 1657
Figure 426. Output stage of capture/compare channel (channel 5, idem ch. 6) .............. 1658
Figure 427. PWM input mode timing ............................................................................. 1660
Figure 428. PWM output mode ...................................................................................... 1662
Figure 429. Edge-aligned PWM waveforms (ARR = 8) ................................................. 1663
Figure 430. Center-aligned PWM waveforms (ARR = 8) ............................................. 1664
Figure 431. Dithering principle ..................................................................................... 1665
Figure 432. Data format and register coding in dithering mode ..................................... 1666
Figure 433. PWM resolution vs frequency ................................................................... 1667
Figure 434. PWM dithering pattern .............................................................................. 1668
Figure 435. Dithering effect on duty cycle in center-aligned PWM mode ..................... 1669
Figure 436. Generation of 2 phase-shifted PWM signals with 50% duty cycle .......... 1671
Figure 437. Combined PWM mode on channel 1 and 3 .............................................. 1672
Figure 438. 3-phase combined PWM signals with multiple trigger pulses per period ... 1673
Figure 439. Complementary output with symmetrical dead-time insertion ............... 1674
Figure 440. Asymmetrical deadtime ............................................................................ 1675
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>442</td>
<td>Dead-time waveforms with delay greater than the negative pulse</td>
<td>1675</td>
</tr>
<tr>
<td>443</td>
<td>Dead-time waveforms with delay greater than the positive pulse</td>
<td>1675</td>
</tr>
<tr>
<td>444</td>
<td>Break and Break2 circuity overview</td>
<td>1678</td>
</tr>
<tr>
<td>445</td>
<td>Various output behavior in response to a break event on tim_brk (OSI = 1)</td>
<td>1680</td>
</tr>
<tr>
<td>446</td>
<td>PWM output state following tim_brk and tim_brk2 assertion (OSI = 1)</td>
<td>1681</td>
</tr>
<tr>
<td>447</td>
<td>PWM output state following tim_brk assertion (OSI = 0)</td>
<td>1682</td>
</tr>
<tr>
<td>448</td>
<td>Output redirection (tim_brk2 request not represented)</td>
<td>1683</td>
</tr>
<tr>
<td>449</td>
<td>Clearing TIMx tim_ocxref</td>
<td>1684</td>
</tr>
<tr>
<td>450</td>
<td>6-step generation, COM example (OSSR = 1)</td>
<td>1685</td>
</tr>
<tr>
<td>451</td>
<td>Example of one pulse mode</td>
<td>1686</td>
</tr>
<tr>
<td>452</td>
<td>Retriggerable one-pulse mode</td>
<td>1688</td>
</tr>
<tr>
<td>453</td>
<td>Pulse generator circuity</td>
<td>1688</td>
</tr>
<tr>
<td>454</td>
<td>Pulse generation on compare event, for edge-aligned and encoder modes</td>
<td>1689</td>
</tr>
<tr>
<td>455</td>
<td>Extended pulselength in case of concurrent triggers</td>
<td>1690</td>
</tr>
<tr>
<td>456</td>
<td>Example of counter operation in encoder interface mode</td>
<td>1692</td>
</tr>
<tr>
<td>457</td>
<td>Example of encoder interface mode with tim_tl1fp1 polarity inverted</td>
<td>1692</td>
</tr>
<tr>
<td>458</td>
<td>Quadrature encoder counting modes</td>
<td>1693</td>
</tr>
<tr>
<td>459</td>
<td>Direction plus clock encoder mode</td>
<td>1694</td>
</tr>
<tr>
<td>460</td>
<td>Directional clock encoder mode (CC1P = CC2P = 0)</td>
<td>1694</td>
</tr>
<tr>
<td>461</td>
<td>Directional clock encoder mode (CC1P = CC2P = 1)</td>
<td>1695</td>
</tr>
<tr>
<td>462</td>
<td>Index gating options</td>
<td>1696</td>
</tr>
<tr>
<td>463</td>
<td>Jittered Index signals</td>
<td>1696</td>
</tr>
<tr>
<td>464</td>
<td>Index generation for IPOS[1:0] = 11</td>
<td>1697</td>
</tr>
<tr>
<td>465</td>
<td>Counter reading with index gated on channel A (IPOS[1:0] = 11)</td>
<td>1697</td>
</tr>
<tr>
<td>466</td>
<td>Counter reading with index ungated (IPOS[1:0] = 00)</td>
<td>1698</td>
</tr>
<tr>
<td>467</td>
<td>Counter reading with index gated on channel A and B.</td>
<td>1698</td>
</tr>
<tr>
<td>468</td>
<td>Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)</td>
<td>1699</td>
</tr>
<tr>
<td>469</td>
<td>Counter reset Narrow index pulse (closer view, ARR = 0x07)</td>
<td>1700</td>
</tr>
<tr>
<td>470</td>
<td>Index behavior in x1 and x2 mode (IPOS[1:0] = 01)</td>
<td>1701</td>
</tr>
<tr>
<td>471</td>
<td>Directional index sensitivity</td>
<td>1701</td>
</tr>
<tr>
<td>472</td>
<td>Counter reset as function of FIDX bit setting</td>
<td>1702</td>
</tr>
<tr>
<td>473</td>
<td>Index blanking</td>
<td>1702</td>
</tr>
<tr>
<td>474</td>
<td>Index behavior in clock + direction mode, IPOS[0] = 1</td>
<td>1703</td>
</tr>
<tr>
<td>475</td>
<td>Index behavior in directional clock mode, IPOS[0] = 1</td>
<td>1703</td>
</tr>
<tr>
<td>476</td>
<td>State diagram for quadrature encoded signals</td>
<td>1704</td>
</tr>
<tr>
<td>477</td>
<td>Up-counting encoder error detection</td>
<td>1705</td>
</tr>
<tr>
<td>478</td>
<td>Down-counting encoder error detection</td>
<td>1706</td>
</tr>
<tr>
<td>479</td>
<td>Encoder mode change with preload transferred on update (SMSPS = 0)</td>
<td>1707</td>
</tr>
<tr>
<td>480</td>
<td>Measuring time interval between edges on three signals</td>
<td>1708</td>
</tr>
<tr>
<td>481</td>
<td>Example of Hall sensor interface</td>
<td>1710</td>
</tr>
<tr>
<td>482</td>
<td>Control circuit in reset mode</td>
<td>1711</td>
</tr>
<tr>
<td>483</td>
<td>Control circuit in Gated mode</td>
<td>1712</td>
</tr>
<tr>
<td>484</td>
<td>Control circuit in trigger mode</td>
<td>1713</td>
</tr>
<tr>
<td>485</td>
<td>Control circuit in external clock mode 2 + trigger mode</td>
<td>1714</td>
</tr>
<tr>
<td>486</td>
<td>General-purpose timer block diagram</td>
<td>1772</td>
</tr>
<tr>
<td>487</td>
<td>Counter timing diagram with prescaler division change from 1 to 2</td>
<td>1777</td>
</tr>
<tr>
<td>488</td>
<td>Counter timing diagram with prescaler division change from 1 to 4</td>
<td>1778</td>
</tr>
<tr>
<td>489</td>
<td>Counter timing diagram, internal clock divided by 1</td>
<td>1779</td>
</tr>
<tr>
<td>490</td>
<td>Counter timing diagram, internal clock divided by 2</td>
<td>1779</td>
</tr>
<tr>
<td>491</td>
<td>Counter timing diagram, internal clock divided by 4</td>
<td>1780</td>
</tr>
<tr>
<td>492</td>
<td>Counter timing diagram, internal clock divided by N</td>
<td>1780</td>
</tr>
<tr>
<td>493</td>
<td>Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)</td>
<td>1781</td>
</tr>
</tbody>
</table>
Figure 550. Index blanking. 1829
Figure 551. Index behavior in clock + direction mode, IPOS[0] = 1. 1830
Figure 552. Index behavior in directional clock mode, IPOS[0] = 1. 1830
Figure 553. State diagram for quadrature encoded signals. 1831
Figure 554. Up-counting encoder error detection. 1832
Figure 555. Down-counting encoder error detection. 1833
Figure 556. Encoder mode change with preload transferred on update (SMSPS = 0). 1834
Figure 557. Control circuit in reset mode. 1836
Figure 558. Control circuit in gated mode. 1837
Figure 559. Control circuit in trigger mode. 1837
Figure 560. Control circuit in external clock mode 2 + trigger mode. 1839
Figure 561. Master/Slave timer example. 1839
Figure 562. Master/slave connection example with 1 channel only timers. 1840
Figure 563. Gating TIM.slv with tim_oc1ref of TIM.mstr. 1841
Figure 564. Gating TIM.slv with Enable of TIM.mstr. 1842
Figure 565. Triggering TIM.slv with update of TIM.mstr. 1843
Figure 566. Triggering TIM.slv with Enable of TIM.mstr. 1843
Figure 567. Triggering TIM.mstr and TIM.slv with TIM.mstr tim_ti1 input. 1844
Figure 568. Basic timer block diagram. 1884
Figure 569. Control circuit in normal mode, internal clock divided by 1. 1885
Figure 570. Counter timing diagram with prescaler division change from 1 to 2. 1887
Figure 571. Counter timing diagram with prescaler division change from 1 to 4. 1887
Figure 572. Counter timing diagram, internal clock divided by 1. 1888
Figure 573. Counter timing diagram, internal clock divided by 2. 1889
Figure 574. Counter timing diagram, internal clock divided by 4. 1889
Figure 575. Counter timing diagram, internal clock divided by N. 1890
Figure 576. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). 1890
Figure 577. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). 1891
Figure 578. Dithering principle. 1892
Figure 579. Data format and register coding in dithering mode. 1892
Figure 580. FCnt resolution vs frequency. 1893
Figure 581. PWM dithering pattern. 1893
Figure 582. General-purpose timer block diagram (TIM9/TIM12). 1904
Figure 583. General-purpose timer block diagram (TIM13/TIM14). 1905
Figure 584. Counter timing diagram with prescaler division change from 1 to 2. 1908
Figure 585. Counter timing diagram with prescaler division change from 1 to 4. 1908
Figure 586. Counter timing diagram, internal clock divided by 1. 1909
Figure 587. Counter timing diagram, internal clock divided by 2. 1910
Figure 588. Counter timing diagram, internal clock divided by 4. 1910
Figure 589. Counter timing diagram, internal clock divided by N. 1911
Figure 590. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded). 1911
Figure 591. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded). 1912
Figure 592. Control circuit in normal mode, internal clock divided by 1. 1913
Figure 593. tim_ti2 external clock connection example. 1913
Figure 594. Control circuit in external clock mode 1  .......................................................... 1914
Figure 595. Capture/compare channel 1 input stage (TIM13/TIM14) ........................................ 1915
Figure 596. Capture/compare channel 1 input stage (TIM9/TIM12) ........................................ 1915
Figure 597. Capture/compare channel 1 main circuit ........................................................... 1916
Figure 598. Output stage of capture/compare channel 1 ........................................................ 1916
Figure 599. PWM input mode timing .......................................................... 1918
Figure 600. Output compare mode, toggle on tim_oc1 ........................................................... 1920
Figure 601. Edge-aligned PWM waveforms (ARR = 8) ......................................................... 1921
Figure 602. Dithering principle ........................................................................ 1922
Figure 603. Data format and register coding in dithering mode .............................................. 1922
Figure 604. PWM resolution vs frequency ........................................................................ 1923
Figure 605. PWM dithering pattern ..................................................................................... 1924
Figure 606. Combined PWM mode on channel 1 and 2 ....................................................... 1926
Figure 607. Example of one pulse mode .............................................................................. 1927
Figure 608. Retriggerable one pulse mode ......................................................................... 1928
Figure 609. Measuring time interval between edges on 2 signals ........................................ 1929
Figure 610. Control circuit in reset mode ........................................................................... 1930
Figure 611. Control circuit in gated mode ......................................................................... 1931
Figure 612. Control circuit in trigger mode ......................................................................... 1931
Figure 613. TIM15 block diagram .................................................................................... 1967
Figure 614. TIM16/TIM17 block diagram ........................................................................... 1968
Figure 615. Counter timing diagram with prescaler division change from 1 to 2 ...................... 1972
Figure 616. Counter timing diagram with prescaler division change from 1 to 4 ...................... 1973
Figure 617. Counter timing diagram, internal clock divided by 1 ......................................... 1974
Figure 618. Counter timing diagram, internal clock divided by 2 ......................................... 1975
Figure 619. Counter timing diagram, internal clock divided by 4 ......................................... 1975
Figure 620. Counter timing diagram, internal clock divided by N ......................................... 1976
Figure 621. Counter timing diagram, update event when ARPE = 0
(TIMx_ARR not preloaded) ..................................................................................... 1976
Figure 622. Counter timing diagram, update event when ARPE = 1
(TIMx_ARR preloaded) ..................................................................................... 1977
Figure 623. Update rate examples depending on mode and TIMx_RCR register settings ........ 1978
Figure 624. Control circuit in normal mode, internal clock divided by 1 ................................. 1979
Figure 625. tim_t2 external clock connection example ...................................................... 1979
Figure 626. Control circuit in external clock mode 1 ........................................................... 1980
Figure 627. Capture/compare channel (example: channel 1 input stage) ............................ 1981
Figure 628. Capture/compare channel 1 main circuit .......................................................... 1981
Figure 629. Output stage of capture/compare channel (channel 1) ....................................... 1982
Figure 630. Output stage of capture/compare channel (channel 2 for TIM15) ....................... 1982
Figure 631. PWM input mode timing .............................................................................. 1985
Figure 632. Output compare mode, toggle on tim_oc1 ........................................................... 1987
Figure 633. Edge-aligned PWM waveforms (ARR = 8) ....................................................... 1988
Figure 634. Dithering principle ......................................................................................... 1989
Figure 635. Data format and register coding in dithering mode ............................................. 1989
Figure 636. PWM resolution vs frequency ......................................................................... 1990
Figure 637. PWM dithering pattern ................................................................................... 1991
Figure 638. Combined PWM mode on channel 1 and 2 ...................................................... 1993
Figure 639. Complementary output with symmetrical dead-time insertion ............................ 1994
Figure 640. Asymmetrical deadtime .................................................................................. 1995
Figure 641. Dead-time waveforms with delay greater than the negative pulse ....................... 1995
Figure 642. Dead-time waveforms with delay greater than the positive pulse ....................... 1995
Figure 643. Break circuitry overview ................................................................................. 1997
Figure 644. Output behavior in response to a break event on tim_brk ................................................. 1999
Figure 645. Output redirection .......................................................... 2001
Figure 646. tim_ocref_clr input selection multiplexer ................................................................. 2002
Figure 647. 6-step generation, COM example (OSSR = 1) .............................................................. 2003
Figure 648. Example of one pulse mode. .......................................................... 2004
Figure 649. Retriggerable one pulse mode ............................................................................... 2006
Figure 650. Measuring time interval between edges on 2 signals .................................................. 2006
Figure 651. Control circuit in reset mode ..................................................................................... 2007
Figure 652. Control circuit in gated mode ...................................................................................... 2008
Figure 653. Control circuit in trigger mode .................................................................................... 2009
Figure 654. LPTIM1/2/3 timer block diagram\(^1\) .......................................................................... 2072
Figure 655. LPTIM4/5 timer block diagram\(^1\) ............................................................................. 2073
Figure 656. Glitch filter timing diagram ....................................................................................... 2077
Figure 657. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) .................................................... 2079
Figure 658. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set) .............................................................................................. 2079
Figure 659. LPTIM output waveform, Continuous counting mode configuration ........... 2080
Figure 660. Waveform generation .......................................................................................... 2081
Figure 661. Encoder mode counting sequence ............................................................................ 2085
Figure 662. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1) ................................................................................................. 2086
Figure 663. Capture/compare input stage (channel 1) ............................................................... 2087
Figure 664. Capture/compare output stage (channel 1) ............................................................. 2087
Figure 665. Edge-aligned PWM mode (PRELOAD = 1) ............................................................... 2089
Figure 666. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) ....................................... 2090
Figure 667. PWM mode with immediate update versus preloaded update ................................ 2091
Figure 668. Watchdog block diagram ....................................................................................... 2121
Figure 669. Window watchdog timing diagram ............................................................................ 2123
Figure 670. Independent watchdog block diagram ..................................................................... 2128
Figure 671. Reset timing due to timeout ...................................................................................... 2130
Figure 672. Reset timing due to refresh in the not allowed area .................................................. 2131
Figure 673. Example of window comparator update ..................................................................... 2133
Figure 674. Independent watchdog interrupt timing diagram .................................................... 2135
Figure 675. Example of early wake-up comparator update .......................................................... 2136
Figure 676. RTC block diagram ............................................................................................... 2144
Figure 677. TAMP block diagram ............................................................................................. 2193
Figure 678. Backup registers protection zones ............................................................................. 2197
Figure 679. Active tamper filtering ............................................................................................. 2203
Figure 680. Block diagram ......................................................................................................... 2236
Figure 681. I²C-bus protocol ........................................................................................................ 2238
Figure 682. Setup and hold timings ............................................................................................. 2240
Figure 683. I²C initialization flow ............................................................................................... 2242
Figure 684. Data reception ......................................................................................................... 2243
Figure 685. Data transmission .................................................................................................... 2244
Figure 686. Slave initialization flow ............................................................................................ 2247
Figure 687. Transfer sequence flow for I²C slave transmitter, NOSTRETCH = 0 ................. 2249
Figure 688. Transfer sequence flow for I²C slave transmitter, NOSTRETCH = 1 ................ 2250
Figure 689. Transfer bus diagrams for I²C slave transmitter (mandatory events only) ........ 2251
Figure 690. Transfer sequence flow for I²C slave receiver, NOSTRETCH = 0 ....................... 2252
Figure 691. Transfer sequence flow for I²C slave receiver, NOSTRETCH = 1 ....................... 2253
Figure 692. Transfer bus diagrams for I²C slave receiver ............................................................
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>750</td>
<td>Transmission using DMA</td>
<td>2453</td>
</tr>
<tr>
<td>751</td>
<td>IrDA SIR ENDEC block diagram</td>
<td>2447</td>
</tr>
<tr>
<td>752</td>
<td>IrDA data modulation (3/16) - normal mode</td>
<td>2444</td>
</tr>
<tr>
<td>753</td>
<td>Transmission using DMA</td>
<td>2449</td>
</tr>
<tr>
<td>754</td>
<td>Reception using DMA</td>
<td>2450</td>
</tr>
<tr>
<td>755</td>
<td>Hardware flow control between 2 USARTs</td>
<td>2450</td>
</tr>
<tr>
<td>756</td>
<td>RS232 RTS flow control</td>
<td>2451</td>
</tr>
<tr>
<td>757</td>
<td>RS232 CTS flow control</td>
<td>2452</td>
</tr>
<tr>
<td>758</td>
<td>Wake-up event verified (wake-up event = address match, FIFO disabled)</td>
<td>2455</td>
</tr>
<tr>
<td>759</td>
<td>Wake-up event not verified (wake-up event = address match, FIFO disabled)</td>
<td>2455</td>
</tr>
<tr>
<td>760</td>
<td>LPUART block diagram</td>
<td>2502</td>
</tr>
<tr>
<td>761</td>
<td>LPUART word length programming</td>
<td>2505</td>
</tr>
<tr>
<td>762</td>
<td>Configurable stop bits</td>
<td>2507</td>
</tr>
<tr>
<td>763</td>
<td>TC/TXE behavior when transmitting</td>
<td>2509</td>
</tr>
<tr>
<td>764</td>
<td>lpuart_ker_ck clock divider block diagram</td>
<td>2513</td>
</tr>
<tr>
<td>765</td>
<td>Mute mode using Idle line detection</td>
<td>2518</td>
</tr>
<tr>
<td>766</td>
<td>Mute mode using address mark detection</td>
<td>2519</td>
</tr>
<tr>
<td>767</td>
<td>Transmission using DMA</td>
<td>2521</td>
</tr>
<tr>
<td>768</td>
<td>Reception using DMA</td>
<td>2522</td>
</tr>
<tr>
<td>769</td>
<td>Hardware flow control between two LPUARTs</td>
<td>2523</td>
</tr>
<tr>
<td>770</td>
<td>RS232 RTS flow control</td>
<td>2523</td>
</tr>
<tr>
<td>771</td>
<td>RS232 CTS flow control</td>
<td>2524</td>
</tr>
<tr>
<td>772</td>
<td>Wake-up event verified (wake-up event = address match, FIFO disabled)</td>
<td>2527</td>
</tr>
<tr>
<td>773</td>
<td>Wake-up event not verified (wake-up event = address match, FIFO disabled)</td>
<td>2527</td>
</tr>
<tr>
<td>774</td>
<td>SPI/I2S block diagram</td>
<td>2559</td>
</tr>
<tr>
<td>775</td>
<td>Full-duplex single master/ single slave application</td>
<td>2562</td>
</tr>
<tr>
<td>776</td>
<td>Half-duplex single master/ single slave application</td>
<td>2563</td>
</tr>
<tr>
<td>777</td>
<td>Simplex single master / single slave application</td>
<td>2564</td>
</tr>
<tr>
<td>778</td>
<td>Master and three independent slaves connected in star topology</td>
<td>2565</td>
</tr>
<tr>
<td>779</td>
<td>Master and three slaves connected in circular (daisy chain) topology</td>
<td>2566</td>
</tr>
<tr>
<td>780</td>
<td>Multimaster application</td>
<td>2567</td>
</tr>
<tr>
<td>781</td>
<td>Scheme of SS control logic</td>
<td>2569</td>
</tr>
<tr>
<td>782</td>
<td>Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)</td>
<td>2569</td>
</tr>
<tr>
<td>783</td>
<td>SS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)</td>
<td>2570</td>
</tr>
<tr>
<td>784</td>
<td>Data clock timing diagram</td>
<td>2572</td>
</tr>
<tr>
<td>785</td>
<td>Data alignment when data size is not equal to 8, 16 or 32 bits</td>
<td>2573</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>789</td>
<td>Packing data in FIFO for transmission and reception at full feature set instance.</td>
<td></td>
</tr>
<tr>
<td>790</td>
<td>TI mode transfer.</td>
<td></td>
</tr>
<tr>
<td>791</td>
<td>Optional configurations of the slave behavior when an underrun condition is detected.</td>
<td></td>
</tr>
<tr>
<td>792</td>
<td>Waveform examples.</td>
<td></td>
</tr>
<tr>
<td>793</td>
<td>Master I2S Philips protocol waveforms (16/32-bit full accuracy).</td>
<td></td>
</tr>
<tr>
<td>794</td>
<td>I2S Philips standard waveforms.</td>
<td></td>
</tr>
<tr>
<td>795</td>
<td>Master MSB Justified 16-bit or 32-bit full-accuracy length.</td>
<td></td>
</tr>
<tr>
<td>796</td>
<td>Master MSB justified 16 or 24-bit data length.</td>
<td></td>
</tr>
<tr>
<td>797</td>
<td>Slave MSB justified.</td>
<td></td>
</tr>
<tr>
<td>798</td>
<td>LSB justified 16 or 24-bit data length.</td>
<td></td>
</tr>
<tr>
<td>799</td>
<td>Master PCM when the frame length is equal the data length.</td>
<td></td>
</tr>
<tr>
<td>800</td>
<td>Slave PCM standard waveforms.</td>
<td></td>
</tr>
<tr>
<td>801</td>
<td>Startup sequence, I2S Philips standard, master.</td>
<td></td>
</tr>
<tr>
<td>802</td>
<td>Startup sequence, I2S Philips standard, slave.</td>
<td></td>
</tr>
<tr>
<td>803</td>
<td>Stop sequence, I2S Philips standard, master.</td>
<td></td>
</tr>
<tr>
<td>804</td>
<td>I^2S clock generator architecture.</td>
<td></td>
</tr>
<tr>
<td>805</td>
<td>Data Format.</td>
<td></td>
</tr>
<tr>
<td>806</td>
<td>Handling of underrun situation.</td>
<td></td>
</tr>
<tr>
<td>807</td>
<td>Handling of overrun situation.</td>
<td></td>
</tr>
<tr>
<td>808</td>
<td>Frame error detection, with FIXCH = 0.</td>
<td></td>
</tr>
<tr>
<td>809</td>
<td>Frame error detection, with FIXCH = 1.</td>
<td></td>
</tr>
<tr>
<td>810</td>
<td>SAI functional block diagram.</td>
<td></td>
</tr>
<tr>
<td>811</td>
<td>Audio frame.</td>
<td></td>
</tr>
<tr>
<td>812</td>
<td>FS role is start of frame + channel side identification (FSDEF = TRIS = 1).</td>
<td></td>
</tr>
<tr>
<td>813</td>
<td>FS role is start of frame (FSDEF = 0).</td>
<td></td>
</tr>
<tr>
<td>814</td>
<td>Slot size configuration with FBOFF = 0 in SAI_xSLOTRE.</td>
<td></td>
</tr>
<tr>
<td>815</td>
<td>First bit offset.</td>
<td></td>
</tr>
<tr>
<td>816</td>
<td>Audio block clock generator overview.</td>
<td></td>
</tr>
<tr>
<td>817</td>
<td>PDM typical connection and timing.</td>
<td></td>
</tr>
<tr>
<td>818</td>
<td>Detailed PDM interface block diagram.</td>
<td></td>
</tr>
<tr>
<td>819</td>
<td>Start-up sequence.</td>
<td></td>
</tr>
<tr>
<td>820</td>
<td>SAI_ADR format in TDM, 32-bit slot width.</td>
<td></td>
</tr>
<tr>
<td>821</td>
<td>SAI_ADR format in TDM, 16-bit slot width.</td>
<td></td>
</tr>
<tr>
<td>822</td>
<td>SAI_ADR format in TDM, 8-bit slot width.</td>
<td></td>
</tr>
<tr>
<td>823</td>
<td>FIFO underrun event.</td>
<td></td>
</tr>
<tr>
<td>824</td>
<td>Example of typical AC’97 configuration on devices featuring at least 2 embedded SAIs (three external AC’97 decoders).</td>
<td></td>
</tr>
<tr>
<td>825</td>
<td>SPDIF format.</td>
<td></td>
</tr>
<tr>
<td>826</td>
<td>XDR register ordering.</td>
<td></td>
</tr>
<tr>
<td>827</td>
<td>Data companding hardware in an audio block in the SAI.</td>
<td></td>
</tr>
<tr>
<td>828</td>
<td>Tristate strategy on SD output line on an inactive slot.</td>
<td></td>
</tr>
<tr>
<td>829</td>
<td>Tristate on output data line in a protocol like I2S.</td>
<td></td>
</tr>
<tr>
<td>830</td>
<td>Overrun detection error.</td>
<td></td>
</tr>
<tr>
<td>831</td>
<td>SPDIFRX block diagram.</td>
<td></td>
</tr>
<tr>
<td>832</td>
<td>SPDIF sub-frame format.</td>
<td></td>
</tr>
<tr>
<td>833</td>
<td>SPDIF block format.</td>
<td></td>
</tr>
<tr>
<td>834</td>
<td>SPDIF Preambles.</td>
<td></td>
</tr>
<tr>
<td>835</td>
<td>Channel coding example.</td>
<td></td>
</tr>
<tr>
<td>836</td>
<td>SPDIFRX decoder.</td>
<td></td>
</tr>
<tr>
<td>837</td>
<td>Noise filtering and edge detection.</td>
<td></td>
</tr>
<tr>
<td>Figure</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>889</td>
<td>Message RAM configuration</td>
<td>2844</td>
</tr>
<tr>
<td>890</td>
<td>Standard message ID filter path</td>
<td>2847</td>
</tr>
<tr>
<td>891</td>
<td>Extended message ID filter path</td>
<td>2848</td>
</tr>
<tr>
<td>892</td>
<td>OTG_FS full-speed block diagram</td>
<td>2898</td>
</tr>
<tr>
<td>893</td>
<td>OTG_FS A-B device connection</td>
<td>2900</td>
</tr>
<tr>
<td>894</td>
<td>OTG_FS peripheral-only connection</td>
<td>2901</td>
</tr>
<tr>
<td>895</td>
<td>OTG_FS host-only connection</td>
<td>2905</td>
</tr>
<tr>
<td>896</td>
<td>SOF connectivity (SOF trigger output to TIM and ITR1 connection)</td>
<td>2909</td>
</tr>
<tr>
<td>897</td>
<td>Updating OTG_HFIR dynamically (RLDCTRL = 1)</td>
<td>2911</td>
</tr>
<tr>
<td>898</td>
<td>Device-mode FIFO address mapping and AHB FIFO access mapping</td>
<td>2912</td>
</tr>
<tr>
<td>899</td>
<td>Host-mode FIFO address mapping and AHB FIFO access mapping</td>
<td>2913</td>
</tr>
<tr>
<td>900</td>
<td>Interrupt hierarchy</td>
<td>2917</td>
</tr>
<tr>
<td>901</td>
<td>Transmit FIFO write task</td>
<td>3009</td>
</tr>
<tr>
<td>902</td>
<td>Receive FIFO read task</td>
<td>3010</td>
</tr>
<tr>
<td>903</td>
<td>Normal bulk/control OUT/SETUP</td>
<td>3011</td>
</tr>
<tr>
<td>904</td>
<td>Bulk/control IN transactions</td>
<td>3015</td>
</tr>
<tr>
<td>905</td>
<td>Normal interrupt IN</td>
<td>3018</td>
</tr>
<tr>
<td>906</td>
<td>Normal interrupt OUT</td>
<td>3023</td>
</tr>
<tr>
<td>907</td>
<td>Isochronous OUT transactions</td>
<td>3025</td>
</tr>
<tr>
<td>908</td>
<td>Isochronous IN transactions</td>
<td>3028</td>
</tr>
<tr>
<td>909</td>
<td>Receive FIFO packet read</td>
<td>3032</td>
</tr>
<tr>
<td>910</td>
<td>Processing a SETUP packet</td>
<td>3034</td>
</tr>
<tr>
<td>911</td>
<td>Bulk OUT transaction</td>
<td>3041</td>
</tr>
<tr>
<td>912</td>
<td>TRDT max timing case</td>
<td>3051</td>
</tr>
<tr>
<td>913</td>
<td>OTG_HS high-speed block diagram</td>
<td>3056</td>
</tr>
<tr>
<td>914</td>
<td>OTG_HS A-B device connection</td>
<td>3058</td>
</tr>
<tr>
<td>915</td>
<td>OTG_HS peripheral-only connection</td>
<td>3059</td>
</tr>
<tr>
<td>916</td>
<td>OTG_HS host-only connection</td>
<td>3063</td>
</tr>
<tr>
<td>917</td>
<td>SOF connectivity (SOF trigger output to TIM and ITR1 connection)</td>
<td>3066</td>
</tr>
<tr>
<td>918</td>
<td>Updating OTG_HFIR dynamically (RLDCTRL = 1)</td>
<td>3069</td>
</tr>
<tr>
<td>919</td>
<td>Device-mode FIFO address mapping and AHB FIFO access mapping</td>
<td>3070</td>
</tr>
<tr>
<td>920</td>
<td>Host-mode FIFO address mapping and AHB FIFO access mapping</td>
<td>3071</td>
</tr>
<tr>
<td>921</td>
<td>Interrupt hierarchy</td>
<td>3075</td>
</tr>
<tr>
<td>922</td>
<td>Transmit FIFO write task</td>
<td>3169</td>
</tr>
<tr>
<td>923</td>
<td>Receive FIFO read task</td>
<td>3170</td>
</tr>
<tr>
<td>924</td>
<td>Normal bulk/control OUT/SETUP</td>
<td>3171</td>
</tr>
<tr>
<td>925</td>
<td>Bulk/control IN transactions</td>
<td>3175</td>
</tr>
<tr>
<td>926</td>
<td>Normal interrupt OUT</td>
<td>3178</td>
</tr>
<tr>
<td>927</td>
<td>Normal interrupt IN</td>
<td>3183</td>
</tr>
<tr>
<td>928</td>
<td>Isochronous OUT transactions</td>
<td>3185</td>
</tr>
<tr>
<td>929</td>
<td>Isochronous IN transactions</td>
<td>3188</td>
</tr>
<tr>
<td>930</td>
<td>Normal bulk/control OUT/SETUP transactions - DMA</td>
<td>3190</td>
</tr>
<tr>
<td>931</td>
<td>Normal bulk/control IN transaction - DMA</td>
<td>3192</td>
</tr>
<tr>
<td>932</td>
<td>Normal interrupt OUT transactions - DMA mode</td>
<td>3193</td>
</tr>
<tr>
<td>933</td>
<td>Normal interrupt IN transactions - DMA mode</td>
<td>3194</td>
</tr>
<tr>
<td>934</td>
<td>Normal isochronous OUT transaction - DMA mode</td>
<td>3195</td>
</tr>
<tr>
<td>935</td>
<td>Normal isochronous IN transactions - DMA mode</td>
<td>3196</td>
</tr>
<tr>
<td>936</td>
<td>Receive FIFO packet read</td>
<td>3202</td>
</tr>
<tr>
<td>937</td>
<td>Processing a SETUP packet</td>
<td>3204</td>
</tr>
<tr>
<td>938</td>
<td>Bulk OUT transaction</td>
<td>3211</td>
</tr>
<tr>
<td>939</td>
<td>TRDT max timing case</td>
<td>3220</td>
</tr>
<tr>
<td>940</td>
<td>UCPD block diagram</td>
<td>3224</td>
</tr>
</tbody>
</table>
Figure 941. Clock division and timing elements ................................................................. 3225
Figure 942. K-code transmission .................................................................................... 3228
Figure 943. Transmit order for various sizes of data ....................................................... 3229
Figure 944. Packet format .............................................................................................. 3230
Figure 945. Line format of Hard Reset ........................................................................... 3230
Figure 946. Line format of Cable Reset ........................................................................... 3231
Figure 947. BIST test data frame ................................................................................... 3232
Figure 948. BIST Carrier Mode 2 frame ......................................................................... 3232
Figure 949. UCPD BMC transmitter architecture ........................................................... 3233
Figure 950. UCPD BMC receiver architecture ............................................................... 3234
Figure 951. Ethernet high-level block diagram ............................................................... 3267
Figure 952. DMA transmission flow (standard mode) ..................................................... 3270
Figure 953. DMA transmission flow (OSP mode) ......................................................... 3272
Figure 954. Receive DMA flow ...................................................................................... 3274
Figure 955. Overview of MAC transmission flow ........................................................ 3278
Figure 956. MAC reception flow .................................................................................. 3280
Figure 957. Packet filtering sequence .......................................................................... 3284
Figure 958. Networked time synchronization ................................................................ 3293
Figure 959. Propagation delay calculation in clocks supporting peer-to-peer path correction ......................................................................................................................... 3294
Figure 960. System time update using fine correction method ........................................ 3304
Figure 961. TCP segmentation offload overview ........................................................... 3321
Figure 962. TCP segmentation offload flow .................................................................. 3322
Figure 963. Header and payload fields of segmented packets ......................................... 3325
Figure 964. Supported PHY interfaces ......................................................................... 3335
Figure 965. SMA Interface block .................................................................................. 3335
Figure 966. MDIO packet structure (Clause 45) ............................................................. 3336
Figure 967. MDIO packet structure (Clause 22) .............................................................. 3337
Figure 968. SMA write operation flow ......................................................................... 3339
Figure 969. Write data packet ...................................................................................... 3340
Figure 970. Read data packet ....................................................................................... 3340
Figure 971. Media independent interface (MII) signals .................................................. 3342
Figure 972. RMII block diagram ................................................................................... 3344
Figure 973. Transmission bit order .............................................................................. 3345
Figure 974. Receive bit order ......................................................................................... 3346
Figure 975. LPI transitions (Transmit, 100 Mbdps) ......................................................... 3354
Figure 976. LPI Tx clock gating (when LPITCSE = 1) .................................................... 3355
Figure 977. LPI transitions (receive, 100 Mbps) ............................................................ 3356
Figure 978. Descriptor ring structure .......................................................................... 3377
Figure 979. DMA descriptor ring .................................................................................. 3378
Figure 980. Descriptor tail pointer example 1 ................................................................ 3379
Figure 981. Descriptor tail pointer example 2 ................................................................ 3379
Figure 982. Transmit descriptor (read format) ............................................................... 3380
Figure 983. Transmit descriptor write-back format ...................................................... 3385
Figure 984. Transmit context descriptor format ......................................................... 3389
Figure 985. Receive descriptor (read format) ............................................................... 3393
Figure 986. Receive descriptor (write-back format) ..................................................... 3395
Figure 987. Receive context descriptor ........................................................................ 3403
Figure 988. Generation of ETH_DMAISR flags .......................................................... 3419
Figure 989. HDMI-CEC block diagram ....................................................................... 3539
Figure 990. Message structure .................................................................................... 3539
Figure 991. Blocks ...................................................................................................... 3540
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>992</td>
<td>Bit timings</td>
<td>3540</td>
</tr>
<tr>
<td>993</td>
<td>Signal free time</td>
<td>3541</td>
</tr>
<tr>
<td>994</td>
<td>Arbitration phase</td>
<td>3541</td>
</tr>
<tr>
<td>995</td>
<td>SFT of three nominal bit periods</td>
<td>3541</td>
</tr>
<tr>
<td>996</td>
<td>Error bit timing</td>
<td>3542</td>
</tr>
<tr>
<td>997</td>
<td>Error handling</td>
<td>3544</td>
</tr>
<tr>
<td>998</td>
<td>TXERR detection</td>
<td>3545</td>
</tr>
<tr>
<td>999</td>
<td>Block diagram of debug infrastructure</td>
<td>3556</td>
</tr>
<tr>
<td>1000</td>
<td>Clock domains of debug infrastructure</td>
<td>3557</td>
</tr>
<tr>
<td>1001</td>
<td>Product life cycle states and debug authentication</td>
<td>3560</td>
</tr>
<tr>
<td>1002</td>
<td>SWD successful data transfer</td>
<td>3563</td>
</tr>
<tr>
<td>1003</td>
<td>JTAG TAP state machine</td>
<td>3564</td>
</tr>
<tr>
<td>1004</td>
<td>Debug and access port connections</td>
<td>3577</td>
</tr>
<tr>
<td>1005</td>
<td>APB-D CoreSight component topology</td>
<td>3590</td>
</tr>
<tr>
<td>1006</td>
<td>Embedded cross trigger</td>
<td>3597</td>
</tr>
<tr>
<td>1007</td>
<td>Mapping of trigger inputs to outputs</td>
<td>3599</td>
</tr>
<tr>
<td>1008</td>
<td>ETF state transition diagram</td>
<td>3629</td>
</tr>
<tr>
<td>1009</td>
<td>Cortex-M7 CoreSight Topology</td>
<td>3699</td>
</tr>
</tbody>
</table>
1 Documentation conventions

1.1 General information

STM32H7Rx/7Sx devices have an Arm® Cortex®-M7 with FPU core.

1.2 List of abbreviations for registers

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

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

---

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

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

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

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

1.4 Availability of peripherals

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

1.5 Availability of security features

For security feature availability refer to the table below:

<table>
<thead>
<tr>
<th>Security feature</th>
<th>STM32H7S3x8</th>
<th>STM32H7S7x8</th>
<th>STM32H7R3x8</th>
<th>STM32H7R7x8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedded flash memory (FLASH):</td>
<td></td>
<td></td>
<td></td>
<td>Available</td>
</tr>
<tr>
<td>– Flash Secure-only area</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security memory management:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Root secure services (RSS)</td>
<td>Available</td>
<td></td>
<td>Available</td>
<td></td>
</tr>
<tr>
<td>On-the-fly encryption / decryption</td>
<td>Available</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STIRoT</td>
<td></td>
<td></td>
<td></td>
<td>Not available</td>
</tr>
<tr>
<td>Cryptographic processor (CRYP and PKA)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash processor (HASH) with PKA verification always available</td>
<td></td>
<td></td>
<td></td>
<td>Available</td>
</tr>
</tbody>
</table>
## 2 Memory and bus architecture

### 2.1 System architecture

The STM32H7Rx/7Sx architecture relies on a Arm Cortex-M7 core optimized for execution by means of Cache, TCM, AXI, and an AHB bus matrix. This architecture also features a 64-bit AXI, an 32-bit multilayer AHB bus matrix, and bus bridges allow interconnection of bus masters with bus slaves, as shown in Table 2.

<table>
<thead>
<tr>
<th>Bus slave / type(1)</th>
<th>Bus master / type(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cortex-M7 - AXIM</td>
<td>Cortex-M7 - AHB</td>
</tr>
<tr>
<td>Cortex-M7 - ITMC</td>
<td>Cortex-M7 - DTIM</td>
</tr>
<tr>
<td>SDMMC1</td>
<td>HPMMA - AXI</td>
</tr>
<tr>
<td>HPMMA - AHB</td>
<td>GPDMA - AHB</td>
</tr>
<tr>
<td>GPDMA - AHB</td>
<td>ICACHE_AHB</td>
</tr>
<tr>
<td>GPUD_AX1</td>
<td>GPUD_AX2</td>
</tr>
<tr>
<td>DCMPP</td>
<td>GFXMMU</td>
</tr>
<tr>
<td>DMA2D</td>
<td>ETH - AHB</td>
</tr>
<tr>
<td>LTDC</td>
<td>SDMMC2 - AHB</td>
</tr>
<tr>
<td>OTG HS - AHB</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2. Bus-master-to-bus-slave interconnect**

<table>
<thead>
<tr>
<th>Interconnect path and type(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ITCM</td>
</tr>
<tr>
<td>DTCM</td>
</tr>
<tr>
<td>FLASH</td>
</tr>
<tr>
<td>AXI SRAM1</td>
</tr>
<tr>
<td>AXI SRAM2</td>
</tr>
<tr>
<td>AXI SRAM3</td>
</tr>
<tr>
<td>AXI SRAM4</td>
</tr>
<tr>
<td>AHB SRAM1/2</td>
</tr>
<tr>
<td>XSPI1</td>
</tr>
<tr>
<td>XSPI2</td>
</tr>
<tr>
<td>FMC</td>
</tr>
<tr>
<td>GFXMMU</td>
</tr>
<tr>
<td>AHB1 peripherals</td>
</tr>
<tr>
<td>APB1 peripherals</td>
</tr>
<tr>
<td>AHB2 peripherals</td>
</tr>
<tr>
<td>APB2 peripherals</td>
</tr>
<tr>
<td>AHB3 peripherals</td>
</tr>
<tr>
<td>AHB4 peripherals</td>
</tr>
<tr>
<td>APB4 peripherals</td>
</tr>
<tr>
<td>AHB5 peripherals</td>
</tr>
<tr>
<td>APB5 peripherals</td>
</tr>
<tr>
<td>Backup RAM</td>
</tr>
</tbody>
</table>

1. **Bold** font type denotes 64-bit bus, plain type denotes 32-bit bus.
2. “X” = access possible, “-” = access not possible.
3. Every transfer can be done in 64 bits
4. Refer to Table 102: Programmed HPDMA1 request for the AHB/APB peripheral request.
The bus matrix provides access from a master to a slave, enabling concurrent access and efficient operation even when several high-speed peripherals work simultaneously. This architecture is shown in the figure below.

**Figure 1. System architecture**
2.1.1 Bus matrices

AXI bus matrix

The multi AXI bus matrix ensures and arbitrates concurrent accesses from multiple masters to multiple slaves. This allows efficient simultaneous operation of high-speed peripherals.

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

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

AHB bus matrices

The AHB bus matrices ensure and arbitrate concurrent accesses from multiple masters to multiple slaves. This allows efficient simultaneous operation of high-speed peripherals.

The arbitration uses a round-robin algorithm.

2.1.2 Bus-to-bus bridges

To allow peripherals with different types of buses to communicate together, there is a number of bus-to-bus bridges in the system.

The AHB/APB bridges allow connecting peripherals on APB1, APB2, APB4, APB5 to AHB1, AHB2, AHB4 and AHB5, respectively. These AHB/APB bridges provide full synchronous interfacing, which allows the APB peripherals to operate with clocks independent of AHB that they connect to.

The AHB/APB bridges also allow APB1, APB2 and APB4 peripherals to connect to GPDMA1 peripheral buses, without transiting through the AHB BUS.

The AHB/APB bridges convert 8-bit / 16-bit APB data to 32-bit AHB data, by replicating it to the three upper bytes / the upper half-word of the 32-bit word.

The AXI bus matrix incorporates AHB/AXI bus bridge functionality on its slave bus interfaces. The AXI/AHB bus bridges on its master interfaces marked as 32-bit in Figure 1: System architecture are outside the matrix.

The Cortex-M7 CPU provides AHB/TCM-bus (ITCM and DTCM buses) translation from its AHBS slave AHB, allowing the HPDMA controller to access the ITCM and DTCM.

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

2.1.3 TCM-buses

The DTCM and ITCM (data and instruction tightly coupled RAMs) are connected through dedicated TCM buses directly to the Cortex-M7 core. The HPDMA controller can access the DTCM and ITCM through AHBS, a specific CPU slave AHB. The DTCM and ITCM are accessed by Cortex-M7 at CPU clock speed, with zero wait states.

2.1.4 CPU buses

Cortex-M7 AXIM bus

The Cortex-M7 CPU uses the 64-bit AXIM bus to access all memories (excluding ITCM, and DTCM)
The AXIM bus connects the CPU to the AXI bus matrix in the core domain.

**Cortex-M7 ITCM bus**

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

**Cortex-M7 DTCM bus**

The Cortex-M7 CPU uses the 2x32-bit DTCM bus for accessing data in the DTCM. The 2x32-bit DTCM bus allows load/load and load/store instruction pairs to be dual-issued on the DTCM memory. It can also fetch instructions.

**Cortex-M7 AHBS bus**

The Cortex-M7 CPU uses the 32-bit AHBS slave bus to allow the HPDMA controller to access the ITCM and the DTCM.

**Cortex-M7 AHBP bus**

The Cortex-M7 CPU uses the 32-bit AHBP bus for accessing AHB1, AHB2, AHB3, AHB4, APB1, APB2 and APB4 peripherals via the AHB bus matrix.

### 2.1.5 Bus master peripherals

**SDMMC1-bus**

The SDMMC1 uses a 32-bit bus, connected to the AXI bus matrix, through which it can access internal AXI SRAM and Flash memories, and external memories through the XSPIs controller and the FMC.

**SDMMC2-bus**

The SDMMC2 uses a 32-bit bus, connected to the AHB bus matrix. Through the system bus matrices, it can access the internal AXI SRAM, SRAM1, SRAM2, SRAM3 SRAM4, backup RAM, Flash memory, and external memories through the XSPIs controller and the FMC.

**HPDMA-bus**

The HPDMA controller has two bus masters: an AXI 64-bit bus, connected to the AXI bus matrix and an AHB 32-bit bus connected to the Cortex-M7 AHBS slave bus.

The HPDMA is optimized for DMA data transfers between memories since it supports linked list transfers that allow performing a chained list of transfers without the need for CPU intervention. Through the system bus matrices and the Cortex-M7 AHBS slave bus, the HPDMA can access all internal and external memories through the CTOSPIs controller and the FMC.

**GPDMA-bus**

The GPDMA controller has two 32-bit AHB bus masters connected to the AHB bus matrix. The memory bus allows the GPDMA data transfers between memories. Through the system bus matrices, the memory bus can access all internal memories except ITCM and DTCM, and external memories through the XSPIs controller and the FMC. It also supports linked list transfers.
The peripheral bus allows the GPDMA data transfers between two peripherals, between two memories or between a peripheral and a memory. Through the system bus matrices, the peripheral bus can access all internal memories except ITCM and DTCM, external memories through the XSPIs controller and the FMC, and all AHB and APB peripherals. A direct access to APB1 and APB2 is available, without passing through AHB1.

**DMA2D-bus (Chrom-Art Accelerator)**

The DMA2D graphics accelerator uses a 64-bit bus, connected to the AXI bus matrix. Through the system bus matrices, internal AXI SRAM, SRAM1, SRAM2, SRAM3, SRAM4 Flash memories and external memories through the XSPIs controller and the FMC.

**LTDC-bus (LCD-TFT controller)**

The LCD-TFT display controller, LTDC, uses a 64-bit bus, connected to the AXI bus matrix, through which it can access internal AXI SRAM and external memories through the XSPIs controller and the FMC.

**Ethernet MAC-bus**

The Ethernet MAC uses a 32-bit bus, connected to the AHB bus matrix. Through the system bus matrices, it can access all internal memories except backup RAM, ITCM, DTCM it can also access external memories, through the XSPIs controller and the FMC.

**USB OTG-bus**

The USBHS1 peripheral uses 32-bit buses, connected to the AHB bus matrix. Through the system bus matrices, it can access all internal memories except backup RAM, ITCM, DTCM and it access the external memories, through the XSPIs controller and the FMC.

### 2.1.6 GPU2D-bus

These buses connect the GPU2D master interfaces to the bus matrix. These buses are used to load/store data from/to the memory. These buses target the GFXMMU in addition to the data memories: internal Flash memory, internal SRAMs (SRAM1, SRAM2, SRAM3 and SRAM4) and external memories through FMC, XSPIs. A 16-Kbyte data cache (ICACHE) is present on the GPU2D M0 bus in order to improve performances.

### 2.1.7 GFXMMU-bus

This bus connects the GFXMMU master interface to the bus matrix. This bus is used only by the GFXMMU to load/store data from/to the memory. This bus targets the data memories: internal Flash memory, internal SRAMs (SRAM1, SRAM2, SRAM3 and SRAM4) and external memories through FMC, XSPIs. The GFXMMU has also a slave bus connection to be accessed by graphical peripheral master buses.

### 2.1.8 Clocks to functional blocks

Upon reset, clocks to blocks such as peripherals and some memories are disabled (except for the SRAM, DTCM, ITCM and Flash memory). To operate a block with no clock upon reset, the software must first enable its clock through RCC_AHBxENR or RCC_APBxENR register, respectively.
2.2 AXI interconnect matrix (AXIM)

2.2.1 AXI introduction

The AXI (advanced extensible interface) interconnect is based on the Arm® CoreLink™ NIC-400 Network Interconnect. The interconnect has eleven initiator ports, or ASIBs (AMBA slave interface blocks), and ten target ports, or AMIBs (AMBA master interface blocks). The ASIBs are connected to the AMIBs via an AXI switch matrix.

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

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

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

2.2.2 AXI interconnect main features

- 64-bit AXI bus switch matrix with eleven ASIBs and ten AMIBs
- AHB/AXI bridge function built into the ASIBs
- concurrent connectivity of multiple ASIBs to multiple AMIBs
- programmable traffic priority management (QoS - quality of service)
- software-configurable via GPV
2.2.3 AXI interconnect functional description

Block diagram

The AXI interconnect is shown in Figure 2.

Figure 2. AXI interconnect
ASIB configuration

*Table 3* summarizes the characteristics of the ASIBs.

<table>
<thead>
<tr>
<th>ASIB</th>
<th>Connected master</th>
<th>Protocol</th>
<th>Bus width</th>
<th>R/W issuing</th>
</tr>
</thead>
<tbody>
<tr>
<td>INI 1</td>
<td>AHB from AHB peripherals</td>
<td>AHB-lite</td>
<td>32</td>
<td>1/1</td>
</tr>
<tr>
<td>INI 2</td>
<td>SDMMC1</td>
<td>AHB-lite</td>
<td>32</td>
<td>1/1</td>
</tr>
<tr>
<td>INI 3</td>
<td>HPDMA</td>
<td>AXI4</td>
<td>64</td>
<td>8/8</td>
</tr>
<tr>
<td>INI 4</td>
<td>Cortex-M7</td>
<td>AXI4</td>
<td>64</td>
<td>7/32</td>
</tr>
<tr>
<td>INI 5</td>
<td>GPU</td>
<td>AHB</td>
<td>64</td>
<td>1/1</td>
</tr>
<tr>
<td>INI 6</td>
<td></td>
<td>AXI4</td>
<td>64</td>
<td>1/2</td>
</tr>
<tr>
<td>INI 7</td>
<td></td>
<td>AXI4</td>
<td>64</td>
<td>1/0</td>
</tr>
<tr>
<td>INI 8</td>
<td>DCMIPP</td>
<td>AXI4</td>
<td>64</td>
<td>0/1</td>
</tr>
<tr>
<td>INI 9</td>
<td>GFXMMU</td>
<td>AXI4</td>
<td>64</td>
<td>1/2</td>
</tr>
<tr>
<td>INI 10</td>
<td>LTDC</td>
<td>AXI4</td>
<td>64</td>
<td>2/0</td>
</tr>
<tr>
<td>INI 11</td>
<td>DMA2D</td>
<td>AXI4</td>
<td>64</td>
<td>2/1</td>
</tr>
</tbody>
</table>

AMIB configuration

*Table 4* summarizes the characteristics of the AMIBs.

<table>
<thead>
<tr>
<th>AMIB</th>
<th>Connected slave</th>
<th>Protocol</th>
<th>Bus width</th>
<th>R/W/Total acceptance</th>
</tr>
</thead>
<tbody>
<tr>
<td>TARG 1</td>
<td>GFXMMU</td>
<td>AXI4</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 2</td>
<td>AHB Sram</td>
<td>AXI4(1)</td>
<td>32</td>
<td>1/1/2</td>
</tr>
<tr>
<td>TARG 3</td>
<td>FMC</td>
<td>AXI4</td>
<td>64</td>
<td>1/1/2</td>
</tr>
<tr>
<td>TARG 4</td>
<td>XSPI1</td>
<td>AXI4</td>
<td>64</td>
<td>1/1/2</td>
</tr>
<tr>
<td>TARG 5</td>
<td>XSPI2</td>
<td>AXI4</td>
<td>64</td>
<td>1/1/2</td>
</tr>
<tr>
<td>TARG 6</td>
<td>AXI SRAM(2)</td>
<td>AXI4</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 7</td>
<td>AXI SRAM(3)</td>
<td>AXI4</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 8</td>
<td>AXI SRAM</td>
<td>AXI4</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 9</td>
<td>AXI SRAM(4)</td>
<td>AXI4</td>
<td>64</td>
<td>2/2/4</td>
</tr>
<tr>
<td>TARG 10</td>
<td>FLASH</td>
<td>AXI4</td>
<td>64</td>
<td>2/2/4</td>
</tr>
</tbody>
</table>

1. Conversion to AHB protocol is done via an AXI/AHB bridge sitting between AXI interconnect and the connected slave.
2. SRAM shared with the ECC (*Table 11: SRAM4 / ECC configurations*).
3. SRAM shared with the DTCM (*SRAM3 / DTCM configurations*).
4. SRAM shared with the ITCM (*SRAM1 / ITCM configurations*).
Quality of service (QoS)

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

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

The QoS values should be programmed according to the latency requirements for the application. Setting a higher priority for an ASIB ensures a lower latency for transactions initiated by the associated bus master. This can be useful for real-time-constrained tasks, such as graphics processing (LTDC, DMA2D). Assigning a high priority to masters that can make many and frequent accesses to the same slave (such as the Cortex-M7 CPU) can block access to that slave by other lower-priority masters.

Global programmer view (GPV)

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

**AXI interconnect - peripheral ID4 register (AXI_PERIPH_ID_4)**

Address offset: 0x1FD0  
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>COUNT4K[3:0]</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>JEP106CON[3:0]</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 31:8 Reserved, must be kept at reset value.
- Bits 7:4 **COUNT4K[3:0]**: Register file size  
  0x0: N/A
- Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
  0x4: Arm®

**AXI interconnect - peripheral ID0 register (AXI_PERIPH_ID_0)**

Address offset: 0x1FE0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>PARTNUM[7:0]</td>
<td>r</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 31:8 Reserved, must be kept at reset value.
- Bits 7:0 **PARTNUM[7:0]**: Peripheral part number bits 0 to 7  
  0x00: Part number = 0x400
**AXI interconnect - peripheral ID1 register (AXI_PERIPH_ID_1)**

Address offset: 0x1FE4  
Reset value: 0x0000 00B4

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity bits 0 to 3  
0x8: Arm® JEDEC code

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

**AXI interconnect - peripheral ID2 register (AXI_PERIPH_ID_2)**

Address offset: 0x1FE8  
Reset value: 0x0000 004B

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]**: Peripheral revision number  
0x2: r0p2

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

Bits 2:0  **JEP106ID[6:4]**: JEP106 Identity bits 4 to 6  
0x3: Arm® JEDEC code
AXI interconnect - component ID0 register (AXI_COMP_ID_0)

Address offset: 0x1FF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  PREAMBLE[7:0]: Preamble bits 0 to 7
0xD: Common ID value

AXI interconnect - component ID1 register (AXI_COMP_ID_1)

Address offset: 0x1FF4
Reset value: 0x0000 00F0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  CLASS[3:0]: Component class
0xF: Generic IP component class
Bits 3:0  PREAMBLE[11:8]: Preamble bits 8 to 11
0x0: Common ID value
### AXI interconnect - component ID2 register (AXI_COMP_ID_2)

**Address offset:** 0x1FF8  
**Reset value:** 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

### AXI interconnect - component ID3 register (AXI_COMP_ID_3)

**Address offset:** 0x1FFC  
**Reset value:** 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[27:20]:** Preamble bits 20 to 27  
0xB1: Common ID value
AXI interconnect - TARG x bus matrix issuing functionality register
(AXI_TARGx_FN_MOD_ISS_BM)

Address offset: 0x1008 + 0x1000 * x, (x = 1 to 10)
Reset value: 0x0000 0000

Bits 31:2  Reserved, must be kept at reset value.
Bit 1  WRITE_ISS_OVERRIDE: Switch matrix write issuing override for target
  0: Normal issuing capability
  1: Set switch matrix write issuing capability to 1
Bit 0  READ_ISS_OVERRIDE: Switch matrix read issuing override for target
  0: Normal issuing capability
  1: Set switch matrix read issuing capability to 1

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

Address offset: 0x1108+ 0x1000 * x, (x = 1 to 10)
Reset value: 0x0000 0000

Bits 31:2  Reserved, must be kept at reset value.
Bit 1  WRITE_ISS_OVERRIDE: Override AMIB write issuing capability
  0: Normal issuing capability
  1: Force issuing capability to 1
Bit 0  READ_ISS_OVERRIDE: Override AMIB read issuing capability
  0: Normal issuing capability
  1: Force issuing capability to 1
### AXI interconnect - TARG x bus matrix functionality 2 register (AXI_TARG2_FN_MOD2)

**Address offset:** 0x3024  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**Bits 31:1** Reserved, must be kept at reset value.  
**Bit 0** **BYPASS_MERGE:** Disable packing of beats to match the output data width. Unaligned transactions are not realigned to the input data word boundary.  
0: Normal operation  
1: Disable packing

### AXI interconnect - TARG x long burst functionality modification register (AXI_TARG2_FN_MOD_LB)

**Address offset:** 0x302C  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**Bits 31:1** Reserved, must be kept at reset value.  
**Bit 0** **FN_MOD_LB:** Controls burst breaking of long bursts  
0: Long bursts can not be generated at the output of the AMIB  
1: Long bursts can be generated at the output of the AMIB
### AXI interconnect - INIx issuing functionality modification register

**AXI_INIx_FN_MOD**

Address offset: 0x41108 + 0x10000 * x, (x = 1 to 11)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>WRITE_ISS_OVERRIDE: Override ASIB write issuing capability</td>
</tr>
<tr>
<td></td>
<td>0: Normal issuing capability</td>
</tr>
<tr>
<td></td>
<td>1: Force issuing capability to 1</td>
</tr>
<tr>
<td>Bit 1</td>
<td>READ_ISS_OVERRIDE: Override ASIB read issuing capability</td>
</tr>
<tr>
<td></td>
<td>0: Normal issuing capability</td>
</tr>
<tr>
<td></td>
<td>1: Force issuing capability to 1</td>
</tr>
</tbody>
</table>

### AXI interconnect - INI x write QoS register (AXI_INIx_WRITE_QOS)

Address offset: 0x41104 + 0x1000 * x, (x = 1 to 11)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:4</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 3:0</td>
<td>AW_QOS[3:0]: Write channel QoS setting</td>
</tr>
<tr>
<td></td>
<td>0x0: Lowest priority</td>
</tr>
<tr>
<td></td>
<td>0xF: Highest priority</td>
</tr>
</tbody>
</table>
AXI interconnect - INI x issuing functionality modification register
(AXI_INIx_FN_MOD2)
Address offset: 0x41024 + 0x1000 * x, (x = 1 to 2)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td><strong>BYPASS_MERGE</strong>: Disable packing of beats to match the output data width. Unaligned transactions are not realigned to the input data word boundary.</td>
</tr>
<tr>
<td></td>
<td>0: Normal operation</td>
</tr>
<tr>
<td></td>
<td>1: Disable packing</td>
</tr>
</tbody>
</table>

AXI interconnect - INI x read QoS register (AXI_INIx_READ_QOS)
Address offset: 0x41100 + 0x1000 * x, (x = 1 to 11)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3:0</td>
<td><strong>AR_QOS[3:0]</strong>: Read channel QoS setting</td>
</tr>
<tr>
<td></td>
<td>0x0: Lowest priority</td>
</tr>
<tr>
<td></td>
<td>0xF: Highest priority</td>
</tr>
</tbody>
</table>
### 2.2.5 AXI interconnect register map

#### Table 5. AXI interconnect register map and reset values

| Offset  | Register name         | Reset value | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|---------|-----------------------|-------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x1FD0  | AXI_PERIPH_ID_4       | 0 0 0 0 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FD4  | AXI_PERIPH_ID_5       | 0 0 0 0 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FD8  | AXI_PERIPH_ID_6       | 0 0 0 0 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FDC  | AXI_PERIPH_ID_7       | 0 0 0 0 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FE0  | AXI_PERIPH_ID_0       | 0 0 0 0 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FE4  | AXI_PERIPH_ID_1       | 1 0 1 1 0 1 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FE8  | AXI_PERIPH_ID_2       | 0 0 1 0 1 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FEC  | AXI_PERIPH_ID_3       | 0 0 0 0 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FF0  | AXI_COMP_ID_0         | 0 0 0 0 0 1 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FF4  | AXI_COMP_ID_1         | 1 1 1 1 0 0 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x1FF8  | AXI_COMP_ID_2         | 0 0 0 0 0 1 |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
### Table 5. AXI interconnect register map and reset values (continued)

| Offset  | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x1FFC  | AXI_COMP_ID_3          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2000 - 0x2004 | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2008  | AXI_TARG1_FN_MOD_LB    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2024  | AXI_TARG1_FN_MOD2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2028  | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x202C  | AXI_TARG1_FN_MOD      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2030 - 0x2104 | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x2108  | AXI_TARG1_FN_MOD      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x210C - 0x3004 | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 5. AXI interconnect register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3008</td>
<td>AXI_TARG2_ FN_MOD ISS_BM</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>0x300C - 0x3020</td>
<td>Reserved</td>
<td></td>
<td>0x3024</td>
<td>AXI_TARG2_ FN_MOD2</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>0x3028</td>
<td>Reserved</td>
<td>0x302C</td>
<td>AXI_TARG2_ FN_MOD LB</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
</tr>
<tr>
<td>0x3030 - 0x3104</td>
<td>Reserved</td>
<td></td>
<td>0x3108</td>
<td>AXI_TARG2_ FN_MOD</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>0x310C - 0x4004</td>
<td>Reserved</td>
<td></td>
<td>0x4008</td>
<td>AXI_TARG3_ FN_MOD ISS_BM</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
</tr>
</tbody>
</table>

Reset value

<table>
<thead>
<tr>
<th>Offset</th>
<th>Reset value</th>
<th>Offset</th>
<th>Reset value</th>
<th>Offset</th>
<th>Reset value</th>
<th>Offset</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x3008</td>
<td></td>
<td>0x300C</td>
<td></td>
<td>0x3024</td>
<td></td>
<td>0x3028</td>
<td></td>
</tr>
<tr>
<td>0x3030</td>
<td></td>
<td>0x3108</td>
<td></td>
<td>0x4008</td>
<td></td>
<td>0x400C</td>
<td></td>
</tr>
</tbody>
</table>
Table 5. AXI interconnect register map and reset values (continued)

| Offset    | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|-----------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x5008    | AXI_TARG4_    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | FN_MOD_       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | ISS_BM        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x500C - 0x6004 | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x6008    | AXI_TARG5_    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | FN_MOD_       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | ISS_BM        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x600C - 0x7004 | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x7008    | AXI_TARG6_    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | FN_MOD_       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | ISS_BM        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x700C - 0x8004 | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x8008    | AXI_TARG7_    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | FN_MOD_       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | ISS_BM        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x800C - 0x8020 | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x8024    | AXI_TARG7_    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | FN_MOD2       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value |               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 5. AXI interconnect register map and reset values (continued)

| Offset   | Register name                        | Reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|--------------------------------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x8028 - | Reserved                             |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x8108   | AXI_TARG7_FN_MOD                     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x810C-  | Reserved                             | 0           | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9008   | AXI_TARG8_FN_MOD_ISS_BM              |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x900C-  | Reserved                             | 0           | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9024   | AXI_TARG8_FN_MOD2                    |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x902C-  | Reserved                             | 0           | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9108   | AXI_TARG8_FN_MOD                     |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x910C-  | Reserved                             | 0           | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |


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

| Offset   | Register name         | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x42024  | AXI_INI1_FN_MOD2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42028  | AXI_INI1_FN_MOD_AHB   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4202C- | Reserved              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x420FC  |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42100  | AXI_INI1_READ_QOS     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | AR_QOS [3:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42104  | AXI_INI1_WRITE_QOS    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | AW_QOS [3:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x42108  | AXI_INI1_FN_MOD       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4210C- | Reserved              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x430FC  |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x43100  | AXI_INI2_READ_QOS     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | AR_QOS [3:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x43104  | AXI_INI2_WRITE_QOS    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | AW_QOS [3:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
### Table 5. AXI interconnect register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>AR_QOS [3:0]</th>
<th>AW_QOS [3:0]</th>
<th>WR_INC_OVERRIDE</th>
<th>RD_INC_OVERRIDE</th>
<th>WRITE_ISS_OVERRIDE</th>
<th>READ_ISS_OVERRIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x43108</td>
<td>AXI_INI2_ FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4310C-0x44020</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44024</td>
<td>AXI_INI3_ FN_MOD2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44028</td>
<td>AXI_INI3_ FN_MOD_AHB</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44100</td>
<td>AXI_INI3_ READ_QOS</td>
<td>AR_QOS [3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44104</td>
<td>AXI_INI3_ WRITE_QOS</td>
<td>AR_QOS [3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44108</td>
<td>AXI_INI3_ FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4410C-0x450FC</td>
<td>Reserved</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x45100</td>
<td>AXI_INI4_ READ_QOS</td>
<td>AR_QOS [3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Start bit</td>
<td>AW_QOS</td>
<td>AR_QOS</td>
<td>AXI_INI4_WRITE_QOS</td>
<td>AXI_INI4_FN_MOD</td>
<td>AXI_INI5_WRITE_QOS</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------</td>
<td>-----------</td>
<td>--------</td>
<td>--------</td>
<td>--------------------</td>
<td>----------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>0x45104</td>
<td>AXI_INI4_WRITE_QOS</td>
<td>[3:0]</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x45108</td>
<td>AXI_INI4_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4510C-0x460FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x46104</td>
<td>AXI_INI5_WRITE_QOS</td>
<td>[3:0]</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x46108</td>
<td>AXI_INI5_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x4610C-0x470FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x47104</td>
<td>AXI_INI6_WRITE_QOS</td>
<td>[3:0]</td>
<td>0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x47108</td>
<td>AXI_INI6_FN_MOD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 5. AXI interconnect register map and reset values (continued)

| Offset   | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x4710C-0x480FC | Reserved              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x48100 | AXI_INI7_READ_QOS    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | AR_QOS [3:0] |    |
|          | Reset value           | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x48104 | AXI_INI7_WRITE_QOS   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x48108 | AXI_INI7_FN_MOD       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x49100 | AXI_INI8_READ_QOS    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x49104 | AXI_INI8_WRITE_QOS   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x49108 | AXI_INI8_FN_MOD       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4A100 | AXI_INI9_READ_QOS    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4A104 | AXI_INI9_WRITE_QOS   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value           | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 5. AXI interconnect register map and reset values (continued)

| Offset    | Register name               | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|-----------|-----------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x4A108   | AXI_INI9_FN_MOD             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4A10C- 0x4B0FC | Reserved                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4B100   | AXI_INI10_READ_QOS          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4B104   | AXI_INI10_WRITE_QOS         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4B108   | AXI_INI10_FN_MOD            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4B10C- 0x4C0FC | Reserved                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C100   | AXI_INI11_READ_QOS          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C104   | AXI_INI11_WRITE_QOS         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x4C108   | AXI_INI11_FN_MOD            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|           | Reset value                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Column headers: Offset, Register name, 31-0, WRITE_ISS_OVERRIDE, READ_ISS_OVERRIDE.
2.3 Memory organization

2.3.1 Introduction

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

The bytes are coded in memory in Little Endian format. The lowest numbered byte in a word is considered the word’s least significant byte and the highest numbered byte the most significant.
2.3.2 Memory map and register boundary addresses

Figure 3. Memory map
All the memory map areas that are not allocated to on-chip memories and peripherals are considered “Reserved”. For the detailed mapping of available memory and register areas, refer to the following table.

The following tables gives, the default memory attribute and the boundary addresses of the peripherals available in the devices.

### Table 6. Memory map and default device memory area attribute

<table>
<thead>
<tr>
<th>Region</th>
<th>Boundary address</th>
<th>Arm® Cortex®-M7</th>
<th>Type</th>
<th>Attributes</th>
<th>Execute never</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>External Devices</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>External Memories</td>
<td>0xD0000000 - 0x9FFFFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xC0000000 - 0xCFFFFFFF</td>
<td>FMC SDRAM Bank1 (or remap FMC NOR/PSRAM/SRAM 1/2 Bank1)</td>
<td>Device</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>0xBFF00000 - 0xBFFFFFFF</td>
<td>GPV/AXIM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xA0000000 - 0xBFFFFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x90000000 - 0x9FFFFFFF</td>
<td>XSPI1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x80000000 - 0x8FFFFFFF</td>
<td>FMC NAND Flash memory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x70000000 - 0x7FFFFFFF</td>
<td>XSPI2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x60000000 - 0x6FFFFFFF</td>
<td>FMC NOR/PSRAM/SRAM 1 Bank1 (or remap of FMC SDRAM Bank1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Peripherals</strong></td>
<td>0x40000000 - 0x5FFFFFFF</td>
<td>Peripherals. See <strong>Table 7: Register boundary addresses</strong>.</td>
<td>Device</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>RAM</strong></td>
<td>0x38801000 - 0x3FFFFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x38800000 - 0x38800FFF</td>
<td>Backup SRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x38008000 - 0x387FFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x30004000 - 0x30007FFF</td>
<td>SRAM2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x30000000 - 0x30003FFF</td>
<td>SRAM1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x24072000 - 0x24071FFF</td>
<td>AXI SRAM shared with ECC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x24040000 - 0x2405FFFF</td>
<td>AXI SRAM shared with DTCM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x24020000 - 0x2403FFFF</td>
<td>AXI SRAM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x24000000 - 0x2401FFFF</td>
<td>AXI SRAM shared with ITCM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x20030000 - 0x23FFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x20000000 - 0x2002FFFF</td>
<td>DTCM - RAM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 6. Memory map and default device memory area attribute

<table>
<thead>
<tr>
<th>Region</th>
<th>Boundary address</th>
<th>Arm® Cortex®-M7</th>
<th>Type</th>
<th>Attributes</th>
<th>Execute never</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>0x1FF20000 - 0x1FFFFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FF00000 - 0x1FF1FFFFF</td>
<td>System flash memory</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x90000000 - 0x1FEFFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FFF800 - 0x08FFFFF</td>
<td>Flash Read Only (ID, Trim, Stack ID)</td>
<td></td>
<td>Write-through cache attribute</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>0x08FFFF400 - 0x08FFFF7FF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF0000 - 0x08FFF3FF</td>
<td>User OTP (Flash)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08010000 - 0x08FFEFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08000000 - 0x0800000FF</td>
<td>User flash memory bank 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x00010000 - 0x07FFFFFFF</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x00000000 - 0x0002FFFFF</td>
<td>ITCM RAM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 7. Register boundary addresses(1)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x58028000 - 0x5BFFFFFFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58027000 - 0x58027FFFF</td>
<td>ECC_DiAG_D2</td>
<td>AHB4</td>
<td>RAMECC registers</td>
</tr>
<tr>
<td>0x58025000 - 0x58026FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58024C00 - 0x58024FFFF</td>
<td>CRC</td>
<td></td>
<td>CRC registers</td>
</tr>
<tr>
<td>0x58024800 - 0x58024BFFFF</td>
<td>PWR</td>
<td></td>
<td>PWR registers</td>
</tr>
<tr>
<td>0x58024400 - 0x580247FFFF</td>
<td>RCC</td>
<td></td>
<td>RCC registers</td>
</tr>
<tr>
<td>0x58024000 - 0x580243FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58023C00 - 0x58023FFFF</td>
<td>GPIOP</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58023800 - 0x58023BFFFF</td>
<td>GPIOO</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58023000 - 0x580233FFFF</td>
<td>GPIOM</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58022000 - 0x58022FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58021C00 - 0x580211FFFF</td>
<td>GPIOH</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58021800 - 0x58021BFFFF</td>
<td>GPIOG</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58021400 - 0x580217FFFF</td>
<td>GPIOF</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58021000 - 0x580213FFFF</td>
<td>GPIOE</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58020C00 - 0x58020FFFF</td>
<td>GPIOD</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58020800 - 0x58020BFFFF</td>
<td>GPIOC</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58020400 - 0x580207FFFF</td>
<td>GPIOB</td>
<td></td>
<td>GPIO registers</td>
</tr>
<tr>
<td>0x58020000 - 0x580203FFFF</td>
<td>GPIOA</td>
<td></td>
<td>GPIO registers</td>
</tr>
</tbody>
</table>
### Table 7. Register boundary addresses\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x58006C00 - 0x5801FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58006800 - 0x58006BFF</td>
<td>DTS</td>
<td></td>
<td>DTS registers</td>
</tr>
<tr>
<td>0x58004C00 - 0x580067FF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58004800 - 0x58004BFF</td>
<td>IWDG</td>
<td></td>
<td>IWDG registers</td>
</tr>
<tr>
<td>0x58004400 - 0x580047FF</td>
<td>TAMP</td>
<td></td>
<td>TAMP registers</td>
</tr>
<tr>
<td>0x58004000 - 0x580043FF</td>
<td>RTC</td>
<td></td>
<td>RTC registers</td>
</tr>
<tr>
<td>0x58003C00 - 0x58003FFF</td>
<td>VREFBUF</td>
<td></td>
<td>VREFBUF registers</td>
</tr>
<tr>
<td>0x58003400 - 0x58003BFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58003000 - 0x580033FF</td>
<td>LPTIM5</td>
<td>APB4</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58002C00 - 0x58002FFF</td>
<td>LPTIM4</td>
<td></td>
<td>LPTIM registers</td>
</tr>
<tr>
<td>0x58002800 - 0x58002BFF</td>
<td>LPTIM3</td>
<td></td>
<td>LPTIM registers</td>
</tr>
<tr>
<td>0x58002400 - 0x580027FF</td>
<td>LPTIM2</td>
<td></td>
<td>LPTIM registers</td>
</tr>
<tr>
<td>0x58001800 - 0x58001BFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58001400 - 0x580017FF</td>
<td>SPI2S</td>
<td></td>
<td>SPI2S registers</td>
</tr>
<tr>
<td>0x58001000 - 0x580013FF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x5800C00 - 0x58000FFF</td>
<td>LPUART1</td>
<td></td>
<td>LPUART registers</td>
</tr>
<tr>
<td>0x58000800 - 0x58000BFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x58000400 - 0x580007FF</td>
<td>SBS</td>
<td></td>
<td>SBS registers</td>
</tr>
<tr>
<td>0x58000000 - 0x580003FF</td>
<td>EXTI</td>
<td></td>
<td>EXTI registers</td>
</tr>
<tr>
<td>0x54000000 - 0x57FFFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

\(^{(1)}\) For more information, refer to the device data sheet.
### Table 7. Register boundary addresses(1) (continued)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x52015400 - 0x53FFFFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x52015000 - 0x520153FF</td>
<td>ICACHE</td>
<td>AHB5</td>
<td>ICACHE registers</td>
</tr>
<tr>
<td>0x52014000 - 0x52014FFFF</td>
<td>GPU2D</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x52010000 - 0x52013FFFF</td>
<td>GFXMMU</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x5200C400 - 0x5200CFFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x5200C000 - 0x520003FF</td>
<td>MCE3</td>
<td>MCE registers</td>
<td></td>
</tr>
<tr>
<td>0x5200BC00 - 0x5200BFFFF</td>
<td>MCE2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x5200B800 - 0x5200BFFFF</td>
<td>MCE1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x5200B400 - 0x5200BFFFF</td>
<td>XSPI1</td>
<td>XSPI registers</td>
<td></td>
</tr>
<tr>
<td>0x5200B000 - 0x5200B3FF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x5200A900 - 0x5200AFFFF</td>
<td>XSPI2</td>
<td>XSPI registers</td>
<td></td>
</tr>
<tr>
<td>0x52009000 - 0x52009FFFF</td>
<td>ECC DIAG_D1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x52008400 - 0x52008FFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x52008000 - 0x520083FF</td>
<td>DLYB SDMMC1</td>
<td>DLYB registers</td>
<td></td>
</tr>
<tr>
<td>0x52007000 - 0x52007FFFF</td>
<td>SDMMC1</td>
<td>SDMMC registers</td>
<td></td>
</tr>
<tr>
<td>0x52006000 - 0x52006FFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x52005000 - 0x52005FFFF</td>
<td>XSPI1</td>
<td>XSPI registers</td>
<td></td>
</tr>
<tr>
<td>0x52004000 - 0x52004FFFF</td>
<td>FMC</td>
<td>FMC register map</td>
<td></td>
</tr>
<tr>
<td>0x52003000 - 0x52003FFFF</td>
<td>JPEG</td>
<td>JPEG codec registers</td>
<td></td>
</tr>
<tr>
<td>0x52002000 - 0x52002FFFF</td>
<td>FLASH</td>
<td>FLASH registers</td>
<td></td>
</tr>
<tr>
<td>0x52001000 - 0x520011FFFF</td>
<td>DMA2D</td>
<td>DMA2D registers</td>
<td></td>
</tr>
<tr>
<td>0x52000000 - 0x520001FFFF</td>
<td>HPDMA1</td>
<td>HPDMA registers</td>
<td></td>
</tr>
<tr>
<td>0x50010000 - 0x51FFFFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x50004000 - 0x5000FFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x50004000 - 0x500043FF</td>
<td>GFXTIM</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x50002400 - 0x50003FFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x50002000 - 0x500023FFFF</td>
<td>DCMIPP</td>
<td>DCMIPP registers</td>
<td></td>
</tr>
<tr>
<td>0x50001400 - 0x50001FFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x50001000 - 0x500013FFFF</td>
<td>LTDC</td>
<td>LTDC registers</td>
<td></td>
</tr>
<tr>
<td>0x50000000 - 0x50000FFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x48004000 - 0x4FFFFFFF</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### Table 7. Register boundary addresses (continued)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x48024000 - 0x4803FFFF</td>
<td>Reserved</td>
<td>AHB3</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48022000 - 0x48023FFF</td>
<td>PKA+RAM</td>
<td></td>
<td>PKA registers</td>
</tr>
<tr>
<td>0x48021400 - 0x48021FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48021000 - 0x480213FFFF</td>
<td>SAES</td>
<td></td>
<td>SAES</td>
</tr>
<tr>
<td>0x48020000 - 0x48020FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48020800 - 0x48020BFFFF</td>
<td>CRYP</td>
<td></td>
<td>CRYP registers</td>
</tr>
<tr>
<td>0x48020400 - 0x480207FFFF</td>
<td>HASH</td>
<td></td>
<td>HASH registers</td>
</tr>
<tr>
<td>0x48020000 - 0x480203FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48004800 - 0x4801FFFF</td>
<td>Reserved</td>
<td>AHB2</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48004400 - 0x480047FFFF</td>
<td>CORDIC</td>
<td></td>
<td>CORDIC registers</td>
</tr>
<tr>
<td>0x48002800 - 0x48002BFFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48000800 - 0x48000BFFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x48000000 - 0x480003FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42006000 - 0x420063FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42005C00 - 0x42005FFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42005400 - 0x420057FFFF</td>
<td>Reserved</td>
<td></td>
<td>SPI/I2S registers</td>
</tr>
<tr>
<td>0x42004C00 - 0x42004FFF</td>
<td>Reserved</td>
<td></td>
<td>TIM9/TIM12 registers (shared with TIM9)</td>
</tr>
<tr>
<td>0x42004800 - 0x42004BFFFF</td>
<td>Reserved</td>
<td>APB2</td>
<td>TIM15 registers</td>
</tr>
<tr>
<td>0x42004400 - 0x420047FFFF</td>
<td>Reserved</td>
<td></td>
<td>SPI/I2S registers</td>
</tr>
<tr>
<td>0x42004000 - 0x420043FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42003000 - 0x420033FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42001400 - 0x420017FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42001000 - 0x420013FFFF</td>
<td>Reserved</td>
<td></td>
<td>USART registers</td>
</tr>
<tr>
<td>0x42000400 - 0x420007FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x42000000 - 0x420003FFFF</td>
<td>Reserved</td>
<td></td>
<td>TIM1 registers</td>
</tr>
</tbody>
</table>
### Table 7. Register boundary addresses\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x400C0000 - 0x41FFFFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40080000 - 0x400BFFFF</td>
<td>OTG FS</td>
<td></td>
<td>OTG_FS registers</td>
</tr>
<tr>
<td>0x40040000 - 0x4007FFFF</td>
<td>OTG HS</td>
<td></td>
<td>OTG_HS registers</td>
</tr>
<tr>
<td>0x40030000 - 0x4003FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x4002F000 - 0x4002FFFF</td>
<td>ADF&amp;</td>
<td>AHB1</td>
<td>ADF registers</td>
</tr>
<tr>
<td>0x40029400 - 0x40022EFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40028000 - 0x400293FF</td>
<td>ETH1</td>
<td></td>
<td>Ethernet registers</td>
</tr>
<tr>
<td>0x40022400 - 0x40027FFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40022000 - 0x400223FF</td>
<td>ADC1/ADC2</td>
<td></td>
<td>ADC registers (for each ADC)</td>
</tr>
<tr>
<td>0x40021000 - 0x40021FFF</td>
<td>GPDMA1</td>
<td></td>
<td>GPDMA registers</td>
</tr>
<tr>
<td>0x40020000 - 0x40020FFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 7. Register boundary addresses\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Boundary address</th>
<th>Peripheral</th>
<th>Bus</th>
<th>Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4000F000 - 0x4001FFFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x4000EC00 - 0x4000EFFF</td>
<td>UCPD1</td>
<td></td>
<td>UCPD registers</td>
</tr>
<tr>
<td>0x4000B400 - 0x4000EBFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x4000AC00 - 0x4000B3FF</td>
<td>CAN Msg RAM</td>
<td></td>
<td>FDCAN registers</td>
</tr>
<tr>
<td>0x4000A800 - 0x4000ABFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x4000A400 - 0x4000A7FF</td>
<td>FDCAN2</td>
<td></td>
<td>FDCAN registers</td>
</tr>
<tr>
<td>0x4000A000 - 0x4000A3FF</td>
<td>FDCAN1</td>
<td></td>
<td>FDCAN registers</td>
</tr>
<tr>
<td>0x40009800 - 0x40009FFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40008400 - 0x400087FF</td>
<td>CR5</td>
<td></td>
<td>CR5 registers</td>
</tr>
<tr>
<td>0x40008000 - 0x400083FF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40007C00 - 0x40007FFF</td>
<td>UART8</td>
<td></td>
<td>UART8 registers</td>
</tr>
<tr>
<td>0x40006C00 - 0x40006FFF</td>
<td>CEC</td>
<td></td>
<td>HDMI-CEC registers</td>
</tr>
<tr>
<td>0x40005C00 - 0x40005FFF</td>
<td>I2C3</td>
<td></td>
<td>I2C registers</td>
</tr>
<tr>
<td>0x40005000 - 0x400053FF</td>
<td>I2C4</td>
<td></td>
<td>I3C registers</td>
</tr>
<tr>
<td>0x40004800 - 0x40004BFF</td>
<td>USART3</td>
<td></td>
<td>USART registers</td>
</tr>
<tr>
<td>0x40000400 - 0x400007FF</td>
<td>SPI2S2</td>
<td></td>
<td>SPI/I2S registers</td>
</tr>
<tr>
<td>0x40003000 - 0x400037FF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40002C00 - 0x40002FFF</td>
<td>WWATCHDOG</td>
<td></td>
<td>WWDG registers</td>
</tr>
<tr>
<td>0x40002800 - 0x40002BFF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x40002400 - 0x400027FF</td>
<td>LPTIM1</td>
<td></td>
<td>LPTIM registers</td>
</tr>
<tr>
<td>0x40001800 - 0x40001BFF</td>
<td>TIM12 (Light)</td>
<td></td>
<td>TIM6/TIM7 registers</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Memory map and register boundary addresses
2.4 Embedded SRAM

The STM32H7Rx/7Sx devices include up to 456 Kbytes of AXI-SRAM mapped onto the AXI bus divided and shared as described in the following section.

- Up to 192 Kbytes of instruction TCM RAM Table 9: SRAM1 / ITCM configurations. This feature can be configured through the ITCM_AXI_SHARE[1:0] option byte in FLASH_OBW2SRP register as described in the FLASH option byte word 2 status register (FLASH_OBW2SR).

- Up to 192 Kbytes of data TCM RAM Table 10: SRAM3 / DTCM configurations. This feature can be configured through the DTCM_AXI_SHARE[1:0] option byte in described in the FLASH option byte word 2 status register (FLASH_OBW2SR).

- Up to 72 Kbytes when ECC is disabled Table 11: SRAM4 / ECC configurations. This feature can be configured through the ECC_ON_SRAM option byte in FLASH_OBW2SRP register as described in the FLASH option byte word 2 status register (FLASH_OBW2SR).

- 16 Kbyte AHB_SRAM1
- 16 Kbyte AHB_SRAM2

The system AHB SRAM can be accessed as bytes, half-words (16-bit units) or words (32-bit units), while the system AXI SRAM can be accessed as bytes, half-words, words or double-words (64-bit units). These memories can be addressed at maximum system clock frequency without wait state.

The AHB masters can read/write-access an SRAM section concurrently with the Ethernet MAC or the USB OTG HS peripheral accessing another SRAM section. For example, the Ethernet MAC accesses the SRAM2 while the CPU accesses the SRAM1, concurrently.

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

- DTCM-RAM on TCM interface is mapped at the address 0x2000 0000 and accessible by Cortex®-M7, and by HPDMA through AHBS slave bus of the Cortex®-M7 CPU. The DTCM-RAM can be used as read-write segment to host critical real-time data (such as stack and heap) for application running on Cortex®-M7 CPU.

- ITCM-RAM on TCM interface mapped at the address 0x0000 0000 and accessible by Cortex®-M7 and by HPDMA through AHBS slave bus of the Cortex®-M7 CPU. The ITCM-RAM can be used to host code for time-critical routines (such as interrupt handlers) that requires deterministic execution.

The backup RAM is mapped at the address 0x3880 0000 and is accessible by most of the system masters. With a battery connected to the VBAT pin, the backup SRAM can be used to retain data during low-power mode (Standby and VBAT mode).
These SRAMs are made of several blocks that are maintained but not usable in Stop mode (AXI SRAM):

- **AXI SRAM**
  - SRAM1 is split in fix allocation and shared part depending on the user option byte define in `FLASH option byte word 2 status register (FLASH_OBW2SR)`: 1x64-Kbyte fix allocation + 2x64 Kbytes shared with the ITCM.
  - SRAM2: 128-Kbyte not shared (no ECC).
  - SRAM3: 128-Kbytes shared by block of 64-Kbytes with DTCM0 and DTCM1. When shared the same amount is affected to each DTCM per 32-Kbytes each. The amount of memory allocated to each DTCM is defined in `FLASH option byte word 2 status register (FLASH_OBW2SR)`.
  - SRAM4: 72-Kbyte (no ECC) coming from SRAM1 (24-Kbytes) and SRAM3 (48-Kbytes) when ECC is disabled `FLASH option byte word 2 status register (FLASH_OBW2SR)`.

- **backup SRAM**
  - The backup SRAM (BKPSRAM) can be retained in all low-power modes and when VDD is off in VBAT mode. Refer to Section 6.4.4: Backup domain for more details.
  - The backup SRAM is protected by the tamper detection circuit, and are erased by hardware in case of tamper detection. They are also erased by hardware in case of a Backup domain reset. Refer to (link to Section 50: Tamper and backup registers (TAMP)) for more details.

The tables below summarize the different configuration supported by each internal SRAM:

### Table 8. SRAMs structure

<table>
<thead>
<tr>
<th>SRAM</th>
<th>STM32H7Rx/7Sx</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM1</td>
<td>From 0 to 128 Kbytes, shared with ITCM</td>
</tr>
<tr>
<td>SRAM2</td>
<td>128 Kbytes no ECC</td>
</tr>
<tr>
<td>SRAM3</td>
<td>From 0 up to 128 Kbytes, shared with DTCM0/1</td>
</tr>
<tr>
<td>SRAM4</td>
<td>From 0 to 72 Kbytes, shared with ECC</td>
</tr>
<tr>
<td>AHB SRAM1/2</td>
<td>16 Kbytes, AHB domain</td>
</tr>
<tr>
<td>BKPSRAM</td>
<td>4 Kbytes</td>
</tr>
</tbody>
</table>

### Table 9. SRAM1 / ITCM configurations

<table>
<thead>
<tr>
<th>Allocation</th>
<th>SRAM1 / ITCM split (Kbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM1 allocation</td>
<td>0</td>
</tr>
<tr>
<td>ITCM allocation</td>
<td>192</td>
</tr>
</tbody>
</table>

### Table 10. SRAM3 / DTCM configurations

<table>
<thead>
<tr>
<th>Allocation</th>
<th>SRAM3 / DTCM split (Kbytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM3 allocation</td>
<td>0</td>
</tr>
<tr>
<td>DTCM0 allocation</td>
<td>96</td>
</tr>
<tr>
<td>DTCM1 allocation</td>
<td>96</td>
</tr>
</tbody>
</table>
128 Kbyte of RAM (SRAM1 for ITCM, SRAM3 for DTCM) can be used either as TCM or as AXI SRAM. This feature can be configured through the ITCM_AXI_SHARED[1:0] and DTCM_AXI_SHARED[1:0] option byte in FLASH_OBW2SRP register, in Section 5: Embedded flash memory (FLASH), as described in Table 12 and Table 14.

<table>
<thead>
<tr>
<th>ITCM_AXI_SHARED[1,0]</th>
<th>ITCM size (Kbyte)</th>
<th>ITCM memory mapping(1)</th>
<th>AXI size (Kbyte)</th>
<th>AXI SRAM1 memory mapping(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00/11</td>
<td>64</td>
<td>0x00000 0000 - 0x0000 FFFF</td>
<td>128</td>
<td>0x2400 0000 - 0x2401 FFFF</td>
</tr>
<tr>
<td>01</td>
<td>128</td>
<td>0x0000 0000 - 0x0001 FFFF</td>
<td>64</td>
<td>0x2401 0000 - 0x2401 FFFF</td>
</tr>
<tr>
<td>10</td>
<td>192</td>
<td>0x0000 0000 - 0x0002 FFFF</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Access to an unmapped area gives a TCM error.
2. Access to an unmapped area gives a bus error.

<table>
<thead>
<tr>
<th>DTCM_AXI_SHARED[1,0]</th>
<th>DTCM size (Kbyte)</th>
<th>DTCM memory mapping(1)</th>
<th>AXI size (Kbyte)</th>
<th>AXI SRAM3 memory mapping(2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00/11</td>
<td>64</td>
<td>0x2000 0000 - 0x2000 FFFF</td>
<td>128</td>
<td>0x2404 0000 - 0x2405 FFFF</td>
</tr>
<tr>
<td>01</td>
<td>128</td>
<td>0x2000 0000 - 0x2001 FFFF</td>
<td>64</td>
<td>0x2404 0000 - 0x2404 FFFF</td>
</tr>
<tr>
<td>10</td>
<td>192</td>
<td>0x2000 0000 - 0x2002 FFFF</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Access to an unmapped area gives a TCM error.
2. Access to an unmapped area gives a bus error.

72 Kbyte of RAM (SRAM4) can be used either by ECC or as AXI SRAM. This feature can be configured through the ECC_ON_SRAM option byte in FLASH_OBW2SRP register as described in Table 14.

<table>
<thead>
<tr>
<th>ECC_ON_SRAM</th>
<th>AXI size (Kbytes)</th>
<th>AXI SRAM4 memory mapping(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>72</td>
<td>0x2406 0000 - 0x2407 FFFF</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Access to an unmapped area gives a bus error.
2.5 Flash memory overview

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

The Flash memory is organized as follows:
- A main memory block divided into sectors.
- An information block:
  - System memory from which the device boots in System memory boot mode
  - Option bytes to configure read and write protection, BOR level, watchdog software/hardware and reset when the device is in Standby or Stop mode.

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

2.6 Boot configuration

In the STM32H7Rx/7Sx, two different boot areas can be selected through the BOOT0 pin as shown in the Table 15.

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>NVSTATE</th>
<th>BOOT0</th>
<th>IROT_SELECT(1)</th>
<th>Boot area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>OPEN</td>
<td>0</td>
<td>-</td>
<td>Boot from the user flash memory at 0x0800 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>-</td>
<td>Boot from the bootloader</td>
</tr>
<tr>
<td>Provisioning</td>
<td>CLOSE</td>
<td>-</td>
<td>Not 0xB4 or H7R</td>
<td>Boot from the RSS in system flash memory Then jump into the bootloader</td>
</tr>
<tr>
<td>Closed/Locked</td>
<td>CLOSE</td>
<td>0xB4</td>
<td>-</td>
<td>Boot from the RSS in system flash memory Then jump into the user flash memory at 0x0800 0000(2)</td>
</tr>
</tbody>
</table>

1. IROT_SELECT is relevant only on H7S.
2. MPU is enabled when executing OEM FW in user flash memory; depending on needed resources, OEM FW should update MPU configuration or disable MPU.

The values on the BOOT0 pin are latched on the 4th rising edge of SYSCLK after reset release. It is up to the user to set the BOOT0 pin after reset.

The BOOT0 pin is re-sampled when the device exits Standby mode. Consequently, the values must be kept in the required Boot mode configuration when the device is in Standby mode.
After startup delay, the selection of the boot area is done before releasing the processor reset.

**Embedded bootloader**

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

- DFU: USB OTG FS (PM12/PM11) in device mode
- USART1: PA10/PA9
- USART2: PA3/PA2
- USART3: PD9/PD8
- UART4: PD0/PD1
- FDCAN2: PB5/PB1
- I2C1: PB8/PB7
- I2C2: PB10/PB11
- I2C3: PA8/PC9
- I3C1: PB8/PB7
- SPI1: PA4/PA5/PA6/PA7
- SPI2: PB15/PB14/PB13/PB12
- SPI3: PC12/PB4/PB3/PA15

For additional information, refer to *Section 8: System configuration, boot and security (SBS) on page 539*. 
3  RAMECC monitoring (RAMECC)

3.1  Introduction

STM32H7Rx/7Sx devices feature a RAM ECC monitoring unit (RAMECC). This provides a means for application software to enable, verify the ECC status, and execute service routines when an error occurs.

3.2  RAMECC main features

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

- 7 ECC bits are added per 32-bit word.
- 8 ECC bits are added per 64-bit word for ITCM-RAM.

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

The RAMECC monitoring unit includes the following features:

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

When a half word is written to an internal SRAM (except DTCM and ITCM) and a reset occurs, this half word is not really written to the SRAM after reset. This due to the ECC behavior.

To ensure the data are effectively written to DTCM and ITCM internal memories, read back the programmed data.

3.3  RAMECC functional description

The ECC is supported by SRAM1, SRAM3, BKPSRAM and TCM. The ECC is enabled / disabled by the user option byte ECC_ON_SRAM in Section 5.9.34: FLASH option byte word 2 status register (FLASH_OBW2SR).

Seven ECC bits are added per 32-bit access, and nine for 64-bit access, allowing two bits for error detection, and one bit for error correction on memory read access.

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

Caution: In case of a byte or half-word write on SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM. The SRAM is then released.
**RAMECC block diagram**

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

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

The following figure describes the implementation of RAM ECC controllers.

![RAM ECC controller implementation schematic](image)

A dedicated RAM ECC monitoring area is defined for each domain (see Section 3.3.2: RAMECC monitor mapping). The RAMECC allows the collection of ECC diagnostic events from each RAM ECC controller and provides a mean for the CPU to verify the ECC status.

*Figure 5* shows the connection schematic between the RAM ECC controller and the RAMECC monitoring unit.
3.3.1 RAMECC internal signals

*Table 16* lists the internal signals that control the RAMECC unit.

**Table 16. RAMECC internal input/output signals**

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ramecc_hclk</td>
<td>Input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>ecc_diag_evtx</td>
<td>Input</td>
<td>ECC diagnostic event generated by RAMx ECC controller x</td>
</tr>
<tr>
<td>ramecc_it</td>
<td>Output</td>
<td>Interrupt generated by the RAMECC monitoring unit when an ECC error is detected.</td>
</tr>
</tbody>
</table>
3.3.2 RAMECC monitor mapping

STM32H7Rx/7Sx devices feature two RAMECC monitoring units. The inputs from the ECC controllers are mapped as described in Table 17. The RAM ECC event monitoring status and configuration registers are described in Section 3.6: RAMECC registers.

Table 17. ECC controller mapping

<table>
<thead>
<tr>
<th>RAMECC units(1)</th>
<th>Monitor number</th>
<th>SRAM ECC event monitoring status and configuration registers</th>
<th>Size in Kbytes</th>
<th>Address Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECC_DIAG_D1 (AXI SRAM)</td>
<td>0</td>
<td>AXI SRAM1 ECC monitoring unit</td>
<td>See Table 9: SRAM1 / ITCM configurations</td>
<td>0x20</td>
</tr>
<tr>
<td>ECC_DIAG_D1 (AXI SRAM)</td>
<td>1</td>
<td>ITCM-RAM ECC monitoring unit</td>
<td>0x40</td>
<td></td>
</tr>
<tr>
<td>ECC_DIAG_D1 (AXI SRAM)</td>
<td>2</td>
<td>DTCM-RAM ECC monitoring unit</td>
<td>D0TCM</td>
<td>See Table 10: SRAM3 / DTCM configurations</td>
</tr>
<tr>
<td>ECC_DIAG_D1 (AXI SRAM)</td>
<td>3</td>
<td>DTCM-RAM ECC monitoring unit</td>
<td>D1TCM</td>
<td>0x80</td>
</tr>
<tr>
<td>ECC_DIAG_D1 (AXI SRAM)</td>
<td>4</td>
<td>AXI SRAM3 ECC monitoring unit</td>
<td>0xA0</td>
<td></td>
</tr>
<tr>
<td>ECC_DIAG_D2 (Backup/Vbat domain)</td>
<td>0</td>
<td>Reserved</td>
<td>0x20</td>
<td></td>
</tr>
<tr>
<td>ECC_DIAG_D2 (Backup/Vbat domain)</td>
<td>1</td>
<td>BKPRAM ECC monitoring unit</td>
<td>4</td>
<td>0x40</td>
</tr>
</tbody>
</table>

1. See Table 7: Register boundary addresses.

Single and double ECC errors

When a single error is detected, it is automatically corrected and the SEDCF bits are set in the RAMECC monitor x status register (ramecc_mxsr). When a double error is detected, the DEDF bit is set in the RAMECC monitor x status register (ramecc_mxsr). An interrupt is generated if enabled by the GECCSEIE and GIE bits in the RAMECC interrupt enable register (ramecc_ier).

For single and double errors:

- An interrupt is generated if enabled by the GECCSEIE and GIE bit in the RAMECC interrupt enable register (ramecc_ier). Single errors cannot be detected when the SEDC bit is set.
- The context: failing address, data, ECC code are stored in the RAMECC monitor x failing address register (ramecc_mxfar), RAMECC monitor x failing data low register (ramecc_mxfdrl), RAMECC monitor x failing data high register (ramecc_mxfdrh) and RAMECC monitor x failing ECC error code register (ramecc_mxfecr) if enabled by the ECCELEN bit in the RAMECC monitor x configuration register (ramecc_mxcf).

Caution: Double errors cannot be detected when the DEDF bit is set.
3.3.3 ECC fault injection test on SRAM1, SRAM3, and TCM

In order to test the ECC a mechanism, allow to test, 1 or 2 bits by word (for single or double error test respectively) and ECC alteration.

- ECCTEA[1:0] = 00 or 11: Normal mode, ECC and DATA memory access granted
- ECCTEA[1:0] = 01: ECC memory access mode granted, data memory access gated
- ECCTEA[1:0] = 10: ECC memory access mode gated, data memory access granted

Example:

initialization
1. Set the normal mode: ECCTEA[1:0] = 00
2. Write 0xCAFEDECA, ECC = 0x19 (computed by ECC module)

data modification
3. Block ECC memory access: ECCTEA[1:0] = 10
4. Grant data modification: Write 0xCAFEDECB (last bit flipped)
5. Return to normal mode: ECCTEA[1:0] = 00
6. READ 0xCAFEDECB with ECC = 19 single error = ’1’ (if enabled a flag is set and an interrupt to the NVIC is generated)

An ECC fault injection test triggers a system break events in TIM1/TIM15/TIM16/TIM17 if one of the following bits is set in the SBS_BKLOCKR register:

- ARAM1ECC_BL
- ARAM3ECC_BL
- ITCMECC_BL
- DTCMECC_BL

This implies that the test must be performed while the PWM outputs of the timers are in idle state.

3.3.4 Read access latency

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

3.4 RAMECC low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. RAMECC interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The content of RAMECC registers is kept. The ECC is functional and ECC error interrupt causes the device to exit from Stop.</td>
</tr>
<tr>
<td>Standby</td>
<td>The RAMECC peripheral is powered down and must be reinitialized after exiting Standby.</td>
</tr>
</tbody>
</table>
3.5 RAMECC interrupts

*Table 19* lists the RAMECC interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit the Sleep/Stop mode</th>
<th>Exit the Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMECC</td>
<td>ECC single error detection and correction</td>
<td>SEDCF</td>
<td>GECCSEIE</td>
<td>Write 0 in SEDCF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>ECC double error detection</td>
<td>DEDF</td>
<td>GIE</td>
<td>Write 0 in DEDF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
### 3.6 RAMECC registers

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

In the registers described below, x refers to:
- SRAM1/2/3/4 when x = 1/2/3/4 respectively
- BKPSRAM when x = 5

#### 3.6.1 RAMECC interrupt enable register (RAMECC_IER)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GECCDEIE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GECCSEIE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GIE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

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

**Bit 3 GECCDEBWIE**: Global ECC double error on byte write (BW) interrupt enable

- When GECCDEBWIE bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a byte write operation to RAM (incomplete word write).
  - 0: no interrupt generated when an ECC double detection error occurs on byte write
  - 1: interrupt generated if an ECC double detection error occurs on byte write

**Bit 2 GECCDEIE**: Global ECC double error interrupt enable

- When GECCDEIE bit is set to 1, an interrupt is generated when an ECC double detection error occurs during a read operation from RAM.
  - 0: no interrupt generated when an ECC double detection error occurs
  - 1: interrupt generated if an ECC double detection error occurs

**Bit 1 GECCSEIE**: Global ECC single error interrupt enable

- When GECCSEIE bit is set to 1, an interrupt is generated when an ECC single error occurs during a read operation from RAM.
  - 0: no interrupt generated when an ECC single error occurs
  - 1: interrupt generated when an ECC single error occurs

**Bit 0 GIE**: Global interrupt enable

- When GIE bit is set to 1, an interrupt is generated when an enabled global ECC error (GECCDEBWIE, GECCDEIE or GECCSEIE) occurs.
  - 0: no interrupt generated when an ECC error occurs
  - 1: interrupt generated when an ECC error occurs
3.6.2 RAMECC monitor x configuration register (RAMECC_MxCR)

Address offset: 0x00 + 0x20 * (x+1), (x = 0 to 4)

Reset value: 0x0000 0000

x is the ECC monitoring unit number

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 31:18</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 17:16 **ECCTEA[1:0]**: ECC Test ECC access

00: inactive
01: write and read access blocked on data memory
10: write and read access blocked on ECC memory
11: inactive

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **ECCDEBWCEN**: ECC double error on byte write (BW) counter enable
When ECCDEBWCEN bit is set to 1, the occurrence counter is incremented when an ECC double detection error occurs during a byte write operation to RAM.
0: no counter increment when an ECC double detection error occurs on byte write
1: counter increment when an ECC double detection error occurs on byte write

Bit 7 **ECCDECEN**: ECC double error counter enable
When ECCDECEN bit is set to 1, the occurrence counter is incremented when an ECC double error occurs during a read operation from RAM.
0: no counter increment when an ECC double error occurs
1: counter increment when an ECC double error occurs

Bit 6 **ECCSECEN**: ECC single error counter enable
When ECCSECEN bit is set to 1, the occurrence counter is incremented when an ECC single error occurs during a read operation from RAM.
0: no counter increment when an ECC single error occurs
1: counter increment when an ECC single error occurs

Bit 5 **ECCELEN**: ECC error latching enable
When ECCELEN bit is set to 1, if an ECC error occurs (both for single error correction or double detection) during a read operation, the context (address, data and ECC code) that generated the error are latched to their respective registers.
0: no error context preserved when an ECC error occurs
1: error context preserved when an ECC error occurs

Bit 4 **ECCDEBWIE**: ECC double error on byte write (BW) interrupt enable
When ECCDEBWIE bit is set to 1, monitor x generates an interrupt when an ECC double detection error occurs during a byte write operation to RAM.
0: no interrupt generated when an ECC double detection error occurs on byte write
1: interrupt generated if an ECC double detection error occurs on byte write
Bit 3 **ECCDEIE**: ECC double error interrupt enable
When ECCDEIE bit is set to 1, monitor x generates an interrupt when an ECC double
detection error occurs during a read operation from RAM.
0: no interrupt generated when an ECC double detection error occurs
1: interrupt generated if an ECC double detection error occurs

Bit 2 **ECCSEIE**: ECC single error interrupt enable
When ECCSEIE bit is set to 1, monitor x generates an interrupt when an ECC single error
occurs during a read operation from RAM.
0: no interrupt generated when an ECC single error occurs
1: interrupt generated when an ECC single error occurs

Bits 1:0 Reserved, must be kept at reset value.
3.6.3 RAMECC monitor x status register
(RAMECC_MxSR)

Address offset: 0x24 + 0x20 * x, (x = 0 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

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

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

3.6.4 RAMECC monitor x failing address register
(RAMECC_MxFAR)

Address offset: 0x28 + 0x20 * x, (x = 0 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **FADD[31:0]**: ECC error failing address
When an ECC error occurs the FADD bitfield contains the address that generated the ECC error.
3.6.5 RAMECC monitor x failing data low register  
(RAMECC_MxFDRL)  
Address offset: 0x2C + 0x20 * x, (x = 0 to 4)  
Reset value: 0x0000 0000

Bits 31:0  FDATAL[31:0]: Failing data low  
When an ECC error occurs the FDATAL bitfield contains the LSB part of the data that  
generated the error. For 32-bit word SRAM, this bitfield contains the full memory word that  
generated the error.

3.6.6 RAMECC monitor x failing data high register  
(RAMECC_MxFDRH)  
Address offset: 0x30 + 0x20 * x, (x = 0 to 4)  
Reset value: 0x0000 0000

Bits 31:0  FDATH[31:0]: Failing data high (64-bit memory)  
When an ECC error occurs the FDATH bitfield contains the MSB part of the data that  
generated the error.
### 3.6.7 RAMECC monitor x failing ECC error code register (RAMECC\_MxFECR)

Address offset: 0x34 + 0x20 * x, (x = 0 to 4)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FEC[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**FEC[15:0]**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

When an ECC error occurs the FEC bitfield contains the ECC failing code that generated the error.
## RAMECC register map

### Table 20. RAMECC register map and reset values

| Offset       | Register name        | Offset reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------------|----------------------|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00         | RAMECC_IER           |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x20 * x    | RAMECC_MxCR         |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24 + 0x20 *(x - 1) | RAMECC_MxSR         |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28 + 0x20 *(x - 1) | RAMECC_MxFAR        | FADD[31:0]         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x2C + 0x20 *(x - 1) | RAMECC_MxFDRL      | FDATAL[31:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x30 + 0x20 *(x - 1) | RAMECC_MxFDRH      | FDATAH[31:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| x34 + 0x20 *(x - 1) | RAMECC_MxFECR      | FEC[31:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|              | Reset value          |                    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
4 System security

The STM32H7Rx/7Sx products are designed with a comprehensive set of security features. This section explains the different security features available on STM32H7Rx/7Sx devices.

4.1 Key security features

- **Secure firmware installation (SFI):** with device unique cryptographic key pair
  - leveraging the on-chip immutable bootloader that supports the download of image through USART, USB, I²C, I³C, SPI, FDCAN and JTAG
- **Boot entry:** the platform allows selection between native immutable root of trust, or proprietary boot entry (in user flash memory).
- **Security services:** The platform comes with native security services. The services are embedded in the system memory to manage root of trust services. Native root of trust takes care of: platform security including secure boot, secure updates of next boot level (uROT: updatable root of trust), secure debug control (debug reopening, regression control). Security services can be personalized for each OEMs. OEM personalization is done thanks to provisioning tools.
- **Temporal isolation:** boot levels are isolated thanks to HDPL (hide protect level) monotonic counter.
- **Secure storage,** featuring:
  - three nonvolatile areas dedicated to secure storage. Areas are protected with HDPL.
  - Battery-powered volatile secure storage, automatically erased in case of tamper
  - Write-only key registers in the AES engines
  - Device 96-bit unique ID
  - Secure storage can rely to SAES engine to encrypt the stored data to benefit of DHUK properties. All data encrypted with the DHUK benefit of temporal isolations (HDPL), RHUK (Root Hardware Unique Key), EPOCH (regression counter) properties.
    a) Data are isolated: each data encrypted/decrypted relying to SAES+DHUK benefit of the variation of the DHUK for each different isolated area (HDPL0, HDPL1, HDPL2)
    b) RHUK: hardware secret nonvolatile unique per device keys (copy protection).
    c) EPOCH: counter incremented each time a regression is done (anti rollback).
- **General purpose cryptographic acceleration:**
  - AES 256-bit engine, supporting ECB, CBC, CTR, GCM and CCM chaining modes
  - Secure AES 256-bit security co-processor, supporting ECB, CBC, CTR, GCM and CCM chaining modes with side-channel counter-measures and mitigations
  - HASH processor, supporting SHA-1 checksums and SHA-2 secure hash (SHA2, SHA2-384, SHA2-512)
  - Public key accelerator (PKA) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits), implementing side-channel counter measures and mitigations when manipulating secrets
  - True random number generator (RNG), NIST SP800-90B pre-certified
• **3 MCEs**: for memory cipher engine of external memories
  – System bus in-line encryption and decryption
  – Crypto: Support block ciphering AES and Noekeon, and chaining modes and stream mode with AES.
  – Automatic key-erase in case of tamper
  – Optimization for octo-SPI/hexa-SPI/FMC data pre-fetching mechanism.

• **New flexible lifecycle scheme**: Allows product maintenance in-the-field (debug re-opening or regressions)

• **Active tamper and protection**: against temperature, voltage and frequency attacks
  – Up to eight active inputs, eight active output tamper pins, available in different power modes (refer to Section 50.4: TAMP low-power modes).

### 4.2 Secure install

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

The confidentiality of the installed images written in the internal flash memory is also protected using the AES.

The SFI native service leverages the following hardware security features:

- secure boot (see Section 4.3)
- temporal isolation using hide protection (see Section 4.5.1)
- secure execution (see Section 4.6)
- secure storage, with associated cryptographic engines (see Section 4.7 and Section 4.8) and flash dedicated areas

Further information can be found in the application note *Overview secure firmware install (SFI)* (AN4992).
4.3 **Secure boot**

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

The STM32H7Sx offers two options using IROT_SELECT option byte for the immutable code to be executed after a reset. The STiRoT takes advantage of the security services already embedded in the STM32. The other option allows OEMs to manage the secure boot path:

- Security services when STiRoT (ST immutable root of trust, code natively present in the system memory) is selected managing the secure boot of the next boot level. STiRoT takes care of the next level integrity, authenticity and updates using configuration done by OEMs. To do so, the H7Sxx STiRoT manages a boot sequence using several system resets to load the first application boot level from external memory to the SRAM, then launch a reset. The loaded image in SRAM is then verified by STiRoT, before the concerned level is launched. STiRoT can also manage the update of such firmware.

- Proprietary boot entry when the full chain is managed by the OEM (called OEM-iROT). The iROT is to be installed in user flash memory with proper security activation (WRP, HDP).

In the devices, the secure boot takes advantage of hardware security features such as:

- temporal isolation using hide protection levels (HDPL) (see [Section 4.5.1: Temporal isolation using secure hide protection (HDP)])
- secure install and update (see [Section 4.2: Secure install] and [Section 4.4: Secure update])
- per-domain secure storage, with associated cryptographic engines if available (see [Section 4.7: Secure storage] and [Section 4.8: Crypto engines])

This section describes the features specifically designed for secure boot.

### 4.3.1 Unique boot entry

The IROT_SELECT option byte allows boot through the STiRoT or OEM-iROT (user flash memory). This selection is possible only for products embedding cryptographic acceleration (STM32H7Sx).

For more information on the boot mechanisms, see [Section 2.6: Boot configuration].
4.3.2 **Security services in system flash memory**

The security services located in the system flash memory, cover the root secure services (RSS), the STiRoT (immutable root of trust), and ST-DA (debug authentication).

The RSS (root secure service) allows the control of ST extensions loaded in SRAM (called RSS-e). One of the RSS-e manages the SFI (secure firmware Install). SFI allows secure and counted installation of OEM firmware in untrusted production environments (such as OEM contract manufacturer).

The STMicroelectronics immutable code also includes secure runtime services that can be called at runtime when a secure application sets the SBS_RSSCMDR register to a non-null value before triggering a system reset. This runtime feature is deactivated when the product state is different to “Provisioning”.

STM32H7Rx/7Sx products offer the following security services:

- **ST-DA (debug authentication control)**: This allows control of the product life cycle for field return. It controls debug reopening or regression following OEM choices. It manages the debug authentication control feature, giving control of the debug reopening and regressions of the product for after sales (field return) of the product.
- **RSS and RSSe SFI**: to handle secure provisioning and secure firmware installation in the user flash memory.

STM32H7Sxx products offer the following security services:

- **STiRoT (immutable root of trust)**. This allows management of the boot of the system, ensuring the secure boot and secure update of the following stage. It handles the secure provisioning, secure boot, secure updates of the first updatable level of the platform. As the product is an “hybrid” between flashless and flash memory, the different boots from STiRoT are based on a multi-reset flow.

For this multi-reset flow, STiRoT secure boot or secure update sequence rely on:

- An OEM-iLoader that have to be installed in the user flash memory. OEM-iLoader is in charge of loading the first boot level from external memory to the SRAM. If OEM-iLoader detect an update (of the first boot level), it loads it to the SRAM. When an image is ready to be processed by STiRoT, a system reset is launched to let the STiRoT manage the proper action.
- OEM-iLoader relies on a library to access the external flash memory (external memory management).
A typical secure boot flow is as follows:

Hypothesis: The IROT_SELECT has been set to STiRoT and the STiRoT has been provisioned. OEM-iLoader code has been installed and write protected in user flash memory. The first updatable boot level has been provisioned in the external memory.

Flow execution:
1. On the STiRoT initial execution, it verifies then launch the OEM-iLoader.
2. Then OEM-iLoader is able to read external memories:
   a) OEM-iLoader checks if an update image is present for the OEM-uROT. If so, it loads it to the SRAM then launch a reset, to let the STiRoT managing the secure controls before install.
   b) If no update image is present then OEM-iLoader loads OEM-uROT from external memory then launch a reset, to let the STiRoT managing the verification of the image before launching it.
4.4 Secure update

The secure update is the capacity of the product to update its firmware, allowing improvement of the product features, or fixing of vulnerabilities.

The product update is considered to be done in almost 2 parts:
- The iROT takes care of the next updatable level (uROT) hosted in external memory.
- The uROT must manage the overall application updates.

When STiRoT is selected, the secure update is executed in several steps. See the previous paragraph for more details.

When OEM-iROT is selected, it is hosted in the internal flash memory. It can be partially or fully immutable. It must manage the external memories, including the hardware setup, the secure boot, the secure update of the next updatable level (uROT).

The update process is performed in a secure way to prevent unauthorized updates and access to confidential on-device data.

4.5 Resource isolation using hide protect levels

In the STM32H7Rx/7Sx devices, the hardware and software resources used for boot can be isolated using the temporal isolation proposed with the HDPL (HiDe protect level) mechanism.

It is based on a monotonic counter, which by definition allows only incrementing of the levels. When the counter is incremented, the resources of the previous levels become hidden (code and data).

A typical usage of the HDPL, is to allocate the HDPL=1 to the iROT. Then the HDPL=2 to the uROT (including OEM-iLoader), then the application is executed in HDPL=3.

4.5.1 Temporal isolation using secure hide protection (HDP)

The embedded flash memory allows definition of an HDP area associated to an HDPL with an 8-Kbyte page granularity (defined through the option bytes). The code executed in this HDP area, with its related data and keys, can be hidden after boot until the next system reset.

The hide protection is related to HDPL=1 (hidden when HDPL becomes equal or greater than 2) when OEM-iROT is selected.

The hide protection is related to HDPL=2 (hidden when HDPL become equal or greater than 3) when STiRoT is selected.

Typically, the activation of an HDP area in user flash memory is related to HDPL1. This means that as soon as the HDPL becomes greater than or equal to 2. Then data read, write and instruction fetch (on the area defined by HDP_AREA_START and HDP_AREA_END in FLASH_HDP option bytes), are denied until the next device reset.

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

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

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

4.6.1 Memory protection unit (MPU)

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

- Cortex-M7 operating mode (privileged, unprivileged)
- data/instruction fetch.

4.6.2 Embedded flash memory write protection

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

Any 8 Kbyte sector can be independently write-protected or unprotected by clearing/setting the corresponding WRPS bit in the FLASH_WRP register.

The embedded flash memory write-protection user option bits can be modified when HDPL=0 or 1 in the SBS_HDPLSR register, using FLASH_WRP.

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

4.6.3 Tamper detection and response

Principle

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

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

See Section 50: Tamper and backup registers (TAMP) for more details.
**Tamper detection sources**

The devices support eight active input/output pins, allowing four independent active-tamper meshes, or up to seven meshes if the same output pin is shared by several input pins (for a total of eight active-tamper I/Os).

The active pins are clocked by the LSE, and are functional in different system operating modes (Run, Sleep, Stop, or Standby), and in VBAT mode. Refer to Section 50.4: TAMP low-power modes for a list of tamper pins and their availability across power modes.

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

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

The internal tamper sources are listed in the table below.

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

\(^1\) LSE missing or over frequency detection (> 2 MHz), Glitch filter (> 2 MHz).
Response to tampers

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

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

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

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

Note: Device secret erase is also triggered by setting the BKERASE bit in the TAMP_CR2 register, or by performing a PRODUCT_STATE regression, as defined in Section 4.9.2. Device secrets are not reset by system reset or when the device wakes up from Standby mode.

Software filtering mechanism

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

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

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

Once the application is notified by the tamper event, it analyzes the situation. There are two possible cases:

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

Note: If the tamper software fails to react to such a tamper flag, an IWDG reset automatically triggers the erase of secrets.
Tamper detection and low-power modes

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

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect on tamper detection features &lt;br&gt;TAMP interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>No effect on tamper detection features, except for level detection with filtering and active tamper modes that remain active only when the clock source is LSE or LSI &lt;br&gt;Tamper events cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>No effect on tamper detection features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI &lt;br&gt;Tamper events cause the device to exit the Standby mode.</td>
</tr>
</tbody>
</table>

### 4.7 Secure storage

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

*Figure 7* shows how a key management-service application can use the AES engine, for example to compute external image decryption keys. Nonvolatile keys are stored in a dedicated flash memory area, including access control (see *Section 4.5.1*), while volatile key storage consists of the battery-powered, tamper-protected SRAM or registers-TAMP.

*Figure 7* also shows keys that are manipulated by software (such as MCE keys), or keys that are managed only by hardware (such as DHUK). More information on those hardware keys can be found in *Section 4.7.1*. 
Details of tamper protection are given in Section 4.6.3.

### 4.7.1 Hardware secret key management

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

There are three different sources of hardware secret keys:

- **DHUK**: derived keys based on 256-bit nonvolatile device unique secret in flash memory. The flash memory provides a value provisioned during product manufacturing (called RHUK). The generation of a DHUK key takes into account the OBK-HDPLx (temporal isolation counter), the EPOCH (regression counter allowing anti-replay protection), and key use state (KMOD).
- **BHK**: 256-bit application key stored in tamper-resistant volatile storage in TAMP. This key is written at boot time, then read/write locked to application until next reset.
- **XORK**: result of a XOR of BHK and DHUK

These keys can be used in the following modes:

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

The devices store a 96-bit ID that is unique to each device (see Section 67.1: Unique device ID register (96 bits)).

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

Alternatively, the 256-bit device unique key (DHUK) can be used (see Section 4.7.1).

4.8 Crypto engines

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

The crypto engines embedded in STM32 reduce weaknesses in the implementation of critical cryptographic functions. They prevent, for example, the use of weak cryptographic algorithms and key sizes. They also enable shorter processing times and lower power consumption when performing cryptographic operations, by offloading those computations from Cortex-M7. This is especially true for asymmetric cryptography.

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

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

4.8.1 Crypto engines features

The table below lists the accelerated cryptographic operations available in the devices. Two AES accelerators are available. One is side-channel attack protected (SAES), while the second is more performance oriented (CRYP).

Note: Additional operations can be added using firmware.

The PKA can accelerate asymmetric crypto operations (like key pair generation, ECC scalar multiplication, point on curve check). See Section 40: Public key accelerator (PKA) for more details.

<table>
<thead>
<tr>
<th>Operations</th>
<th>Algorithm</th>
<th>Specification</th>
<th>Key lengths (in bit)</th>
<th>Modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get entropy</td>
<td>RNG</td>
<td>NIST SP800-90B(1)</td>
<td>N/A</td>
<td>Software and hardware(2) modes running in parallel</td>
</tr>
<tr>
<td>Encryption, decryption</td>
<td>AES</td>
<td>FIPS PUB 197</td>
<td>128, 256</td>
<td>ECB, CBC, CTR(3)</td>
</tr>
<tr>
<td>Authenticated encryption or decryption</td>
<td>AES</td>
<td>NIST SP800-38C</td>
<td>128, 256</td>
<td>GCM, CCM</td>
</tr>
<tr>
<td>Cipher-based message authentication code</td>
<td>AES</td>
<td>NIST SP800-38D</td>
<td>128, 256</td>
<td>GMAC</td>
</tr>
</tbody>
</table>
Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

4.8.2 Secure AES co-processor (SAES)

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

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

As shown in Section 4.7, the SAES can be used for extra-secure on-chip storage for sensitive information.

For more information, refer to Section 36: Secure AES coprocessor (SAES).

4.8.3 Memory cipher engine (MCE)

Three MCEs propose on-the-fly encryption and decryption on external non volatile or volatile memories. Up to 4 regions can be defined on each MCE, with a granularity of 4 kbytes. Access filtering can be applied on different regions (privilege, write).

When a tamper event is confirmed in TAMP, all MCE keys are erased.
4.9 **Product life cycle**

The product life cycle of the STM32H7Rx/7Sx is the result of combining the following settings:

- the flash interface state: `FLASH_NVSRP -> NVSTATE = OPEN /CLOSE`
- the option byte: `FLASH_ROTSR->OEM_PRVD = iROT- provisioned or not`
- the option byte: `FLASH_ROTSR->DBG_AUTH = select the debug authentication control method, between locked, ECDSA (certificates), and password.`

![Figure 8. Product life-cycle security](MSv55538V2)
The features described below contribute to secure the device life-cycle.

### 4.9.1 Product configurations and security services

The product is provisioned with ST native security services. However, the product can be configured to leave full control of the boot chain to OEMs.

This is done by means of the unique boot entry (FLASH_ROTSRP: IROT_SELECT) option byte. This allows selection between the security services (STiRoT: ST immutable root of trust), or OEM boot implementation (OEM-iROT: OEM immutable root of trust) to be installed in the user part of the flash memory. The product boots on the STiRoT only when the IROT_SELECT is set to STiRoT and when the product state is Closed or Locked.

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

STiRoT and OEM-iROT are only available on products embedding cryptographic acceleration (STM32H7Sx).

#### Table 24. Product life-cycle states

<table>
<thead>
<tr>
<th>PRODUCT_STATE</th>
<th>Selecting PRODUCT_STATE</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>NVSTATE = OPEN</td>
<td>This state allows product development, as it provides the code debug. Using the boot pin allows the bootloader to be launched.</td>
</tr>
<tr>
<td>Provisioning</td>
<td>NVSTATE = CLOSE</td>
<td>This state allows product provisioning (partial or full). It allows launching of secure firmware Install, or bootloader to provision the product. Boot from SRAM is not permitted.</td>
</tr>
<tr>
<td></td>
<td>OEM_PROVD=0xB4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBG_AUTH = Not set (0xB4, 0x51, 0x8A)</td>
<td></td>
</tr>
<tr>
<td>Closed</td>
<td>NVSTATE = CLOSE</td>
<td>This state considers that the product configuration is finalized. It allows support of debug authentication for in-the-field repair (read the dedicated application note).</td>
</tr>
<tr>
<td></td>
<td>OEM_PROVD=0xB4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBG_AUTH=0x51 or 0x8A</td>
<td></td>
</tr>
<tr>
<td>Locked</td>
<td>NVSTATE = CLOSE</td>
<td>This state considers that the product configuration is finalized. The debug authentication is not permitted. The product is definitively in this state.</td>
</tr>
<tr>
<td></td>
<td>OEM_PROVD=0xB4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DBG_AUTH=0xB4</td>
<td></td>
</tr>
</tbody>
</table>
4.9.2 Recommended product settings

To ease the product maintenance (in-the-field), we recommend taking advantage of the called debug authentication control feature. This enables product maintenance by activating debug, and makes regression management possible, while considering the security of sensitive information.

This implies setting the product state to provisioning state then provision the DA-config and selecting the final product state (closed)

A dedicated application note is available to help when configuring the debug authentication control of the platform.

4.10 System memory

4.10.1 Introduction

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

4.10.2 RSS user functions

The RSS provides runtime services thanks to RSS library, whose functions are exposed to user within the CMSIS device header file provided by the STM32CubeH7RS firmware package (see AN5935 “Getting started with MCU hardware development for STM32H7R3/7S3 and STM32H7R7/7S7 product lines” for more details).

RSS provides services through the RSSLIB library, dedicated to secure user firmware. Before calling RSSLIB functions, if MPU is enabled, the secure user firmware must:

- Define a MPU region to have read access on this location, starting from RSSLIB_SYS_FLASH_PFUNC_START (0x1FF1FD4C), up to RSSLIB_SYS_FLASH_PFUNC_END (0x1FF1FD78). These addresses are provided within the CMSIS device header file.
- Define a MPU region to allow execution of RSSLIB FW, starting from 0x1FF16800 up to 0x1FF17FFF.

<table>
<thead>
<tr>
<th>C defined macro</th>
<th>Location in system memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSSLIB_PFUNC</td>
<td>0x1FF1FD4C</td>
</tr>
<tr>
<td>&quot;&quot;</td>
<td>&quot;&quot;</td>
</tr>
</tbody>
</table>
**RSSLIB**

The user firmware calls RSSLIB functions using `RSSLIB_PFUNC` C defined macro, which points to a location within non-secure system memory.

**Table 26. RSS lib interface functions**

<table>
<thead>
<tr>
<th>Library</th>
<th>Callable function</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>RSSLIB_PFUNC</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>SetSecOB(1)</code></td>
</tr>
<tr>
<td></td>
<td><code>GetRssStatus(1)</code></td>
</tr>
<tr>
<td></td>
<td><code>SetProductState(1)</code></td>
</tr>
<tr>
<td></td>
<td><code>DataProvisioning(1)</code></td>
</tr>
<tr>
<td></td>
<td><code>JumpHDPLvl2</code></td>
</tr>
<tr>
<td></td>
<td><code>JumpHDPLvl3</code></td>
</tr>
<tr>
<td></td>
<td><code>GetProductState(1)</code></td>
</tr>
</tbody>
</table>

1. Functions used for OEM personalization done using provisioning tools.

**SetSecOB**

Prototype:

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

User code function call example:

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

Arguments:

- **ObAddr**:  
  - Input parameter, address of the secure OB to configure, must be one of:  
    `(FLASH->WRPSRP)`  
    `(FLASH->HDPSRP)`  
    `(FLASH->ROTSRP)`
- **ObMask**:  
  - Input parameter, mask of the OB part to be updated (example `0x000000FF` for updating WRP OB).
- **ObValue**:  
  - Input parameter, OB value to be applied.
- **ObPos**:  
  - Input parameter, OB bit shift to be applied on ObValue (`ObValue<<ObPos` must be aligned on ObMask).

The user calls `SetSecOB` to configure secure OB, the configuration is done by RSS after reset. This function is only available in provisioning product state.

On successful call, the function does not return and does not push LR onto the stack.

In case of failure (bad product state), `SetSecOB` returns `0xF5F5F5F5`. 

---

RM0477 Rev 6 193/3791
To retrieve OB change status (after RSS execution), need to call GetRssStatus function or read back the OB (for example read FLASH_WRPSR register).

**GetRssStatus**

Prototype:
```
uint32_t GetRssStatus(void)
```

User code function call example:
```
RSSLIB_PFUNC->GetRssStatus();
```

Arguments: None

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

Returned values:
- 0xEAEEAEAEA Success
- 0xF5F5F5F5 Error
- 0xF5F5E0E0 Bad input address range
- 0xF5F50E0E Bad size
- 0xF5F5E0EE Bad encryption value in DoEncryption
- 0xF5F50880 Hardware cryptography not available (H7R)
- 0xF5F50EE0 Encryption error
- 0xF5F50808 Programming error
- 0xF5F50008 Bad value in Destination

**SetProductState**

Prototype:
```
uint32_t SetProductState(uint32_t ProductState)
```

User code function call example:
```
RSSLIB_PFUNC->SetProductState((uint32_t) ProductState);
```

Arguments:
- **ProductState:**
  - Input parameter, STM32 product state to reach, must be one of:
    - 0x17: Provisioning product state, allowed when current product state is Open.
    - 0x72: Closed product state, allowed when current product state is Provisioning and Data provisioning has been made.
    - 0x5C: Locked product state, allowed when current product state is Provisioning.

User calls SetProductState to change STM32 product state, the configuration is done by RSS after reset.

On successful call, the function does not return and does not push LR onto the stack.

To retrieve new product state (after RSS execution), need to call **GetProductState** function.

In the case of failure (bad product state), SetProductState returns 0xF5F5F5F5.
DataProvisioning

Prototype:

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

User code function call example:

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

Arguments:

- `pConfig`: input parameter. `RSSLIB_DataProvisioningConf_t`

C structure definition is described below:

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

Structure elements:

- **pSource**: Provides the address of data to be provisioned. Must be within SRAM3 address range.
- **Destination**: Provides the destination information (where to provision the data).
  - 0xDADAHHII:
    - II provides OBKey starting index.
    - HH provides OBKey HDPL.
    - DADA provides Debug Authentication configuration, only relevant when OBKey HDPL0.
- **Size**: Provides the size of data to be provisioned (the number of bytes, must be a multiple of 32).
- **DoEncryption**: Notifies the DataProvisioning function if it must encrypt or not data within OBKeys.
  - DoEncrypt can be either 0xF5F5A0AAU (mandatory on STM32H7Sx) or 0xCACA0AA0U (mandatory on STM32H7Rx)
  - 0xF5F5A0AAU: notifies the DataProvisioning to encrypt data with relevant AHK before programming it within OBKeys. AHK is selected according to the `Destination` value.
  - 0xCACA0AA0U: notifies DataProvisioning to program in clear data within OBKeys.
- **Crc**: CRC over full source data buffer, `pConfig->Destination` value, `pConfig->Size` value and finally `pConfig->DoEncryption` value.

CRC computation uses CRC-32 (Ethernet):
- CRC polynomial: 0x04C11DB7U
- Initial value: 0xFFFFFFFFU
Returned values
0xEAEAEAEAU  Success
0xF5F5F5F5U  Error: bad product state
DataProvisioning only available in Open product state (encryption not supported) and Provisioning product state (encryption supported only on STM32H7Sx).
0xF5F5E0EU  Error: bad input address range.
  pConfig or pSource are not in SRAM3
0xF5F5E0EU  Error: bad provisioning size
  – Size is not a multiple of 32
  – OBKey starting index + Size does not fit within an OBKey section
0xF5F808U   Error: computed CRC is not the expected one provided within pConfig->CRC
0xF5F5E0EU  Error: wrong DoEncryption parameter value
0xF5F0808U  Error: OBKeys programming error
0xF5F5008U  Error: bad Destination information
  – OBKey index not in the range 0 to 31
  – OBKey HDPL not in the range 0 to 2 (representing HDPL0, HDPL1 and HDPL2)
  – On OBKey HDPL0, debug authentication not one of ECDSA, PASSWORD or LOCKED

DataProvisioning receives in input a data buffer and programs it within OBKeys. A CRC prevents any data and parameters tampering issue.

When requested through the pConfig->DoEncryption parameter, DataProvisioning encrypts data before programming them within OBKeys.

DataProvisioning uses AES CBC 128 bits with:
• IV: defined as (using C definition format)
  
  uint32_t IV = {0x8001D1CEU, 0xD1CED1CEU, 0xD1CE8001U, 0xCED1CED1U};
• Key: AHK corresponding to the targeted OBKeys HDPL that is randomly generated during the DA (HDPL0) provisioning that must be done first.

Note: Data encryption within OBkeys is supported only by STM32H7S devices.

JumpHDPLv2

Prototype:
uint32_t JumpHDPLv2(uint32_t VectorTableAddr, uint32_t MPUIndex)

User code function call example:
RSSLIB_PFUNC->JumpHDPLv2((uint32_t)NextVectorTableAddr, 1U);

Arguments:
• VectorTableAddr:
  – Input parameter, address of the next vector table to apply
  – The vector table format is the one used by the Cortex-M7 core
• MPUIndex:
  – Input parameter, MPU region index, (index 0 to 31 allowed). Caller function must define (but keep disabled) the corresponding MPU region before calling
JumpHDPLvl2. The function enables the MPU region before jumping to the reset handler of the vector table. The vector table reset handler function belongs to the MPU region.

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

After closing HDPL1, JumpHDPLvl2 enables the MPU region provided as input parameter. Once the MPU is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by it. JumpHDPLvl2 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In case of failure (bad input parameter value), JumpHDPLvl2 returns 0xF5F5F5F5F5.

JumpHDPLvl3

Secure attribute: Secure callable function.

Prototype:

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

User code function call example:

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

Arguments:

- **VectorTableAddr:**
  - Input parameter, address of the next vector table to apply.
  - The vector table format is the one used by the Cortex-M7 core.
- **MPUIndex:**
  - Input parameter, MPU region index, (index 0 to 31 allowed). Caller function must define but keep disabled the corresponding MPU region before calling JumpHDPLvl3. The function enables the MPU region before jumping to the reset handler of the vector table, whose function belongs to the MPU region.

User calls JumpHDPLvl3 to close user Flash HDPL1 and HDPL2 areas by incrementing HDPL up to 3, then jump to the reset handler embedded within the vector table, whose address is passed as input parameter.

After closing HDPL1/2, JumpHDPLvl3 enables the MPU region provided as input parameter. Once the MPU region is enabled, the function sets the SP to the address provided by the passed vector table, and jumps to the reset handler function supported by the vector table. JumpHDPLvl3 does not set the new vector table.

On successful execution, the function does not return and does not push LR onto the stack.

In the case of failure (bad input parameter value), JumpHDPLvl3 returns 0xF5F5F5F5.
GetProductState

Prototype:
uint32_t GetProductState(void)

User code function call example:
RSSLIB_PFUNC->GetProductState();

Arguments: None

User calls GetProductState to get STM32 product state.

In case of failure (bad product configuration), GetProductState returns 0xF5F5F5F5 (Bad state).

On successful call, the function return one of the following product state:
  • 0x39: Open product state.
  • 0x17: Provisioning product state.
  • 0x72: Closed product state.
  • 0x5C: Locked product state.
5 Embedded flash memory (FLASH)

5.1 Introduction

The embedded flash memory (FLASH) manages the accesses to 196 Kbytes of embedded non-volatile memory (NVM) split into different functional areas such as the immutable root of trust, the STM32 secure firmware install (SFI) or the application’s one-time-programmable information.

FLASH implements the read, program and erase operations, error corrections as well as various integrity and confidentiality protection mechanisms associated to those various functional areas.

The embedded flash memory manages the automatic loading of non-volatile user option bytes at power-on reset, and implements the dynamic update of those options. Those options includes non-volatile application keys that are protected by the device HDPL counter.

5.2 FLASH main features

- 64 Kbytes of non-volatile user flash memory
- Flash memory read operations supporting multiple length (64 bits, 32 bits, 16 bits or one byte)
- Flash memory programming by 128 bits (user area) and 16 bits (OTP area)
- 8 Kbytes sector erase, and bank erase (conditioned)
- Error Code Correction (ECC): one error detection/correction or two error detections per 128-bit flash word using 9 ECC bits
- Cyclic redundancy check (CRC) hardware module
- User configurable non-volatile option byte words
- Up to 1 Kbyte of protected non-volatile option byte keys, readable either by software or usable as a secret AES key through SAES peripheral (as AHK keys)
- Non-volatile security life cycle management of the device
  - Includes controlled security regression with RAM and flash automatic erase
- Temporal isolation enforcement based on application HDPL stored in SBS register
  - Securing one user flash area, and a set of option byte keys
  - Unique boot entry enforcement on a closed device
- 8 Kbytes sector write-protection (WRPS)
- 512 words of 16-bit of one-time programmable (OTP) reserved for user information
- 512-byte read-only area reserved to STMicroelectronics information
- Read command queue to streamline flash operations
- Support for ST immutable Root of Trust in system flash (STM32H7S only)
- Non-volatile 256-bit seed to compute derived root hardware keys (DHUK) in SAES peripheral
- Non-volatile key storage revocation counter (EPOCH)
5.3 **FLASH functional description**

5.3.1 **FLASH block diagram**

*Figure 9* shows the embedded flash memory block diagram, with its interaction with the rest of the device.

*Figure 9. FLASH block diagram*

For more information on **RCC**, **SBS** (with its debug management) and **SAES**, refer to their dedicated chapters.
5.3.2  FLASH internal signals

*Table 27* describes a list of the useful to know internal signals available at embedded flash memory level. These signals are not available on the microcontroller pads.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type (1)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>po_rst</td>
<td>Input</td>
<td>Power on reset</td>
</tr>
<tr>
<td>flash_rst</td>
<td>Input</td>
<td>Embedded flash reset</td>
</tr>
<tr>
<td>flash_aclk</td>
<td>AXI clock of system bus</td>
<td></td>
</tr>
<tr>
<td>flash_it</td>
<td>Output</td>
<td>Embedded flash memory interrupt request</td>
</tr>
<tr>
<td>keybus</td>
<td>Input/Output</td>
<td>Digital interface to transfer some option byte keys to SAES peripheral</td>
</tr>
<tr>
<td>rhuk</td>
<td>Output</td>
<td>256-bit hardware unique key to SAES peripheral</td>
</tr>
<tr>
<td>epoch</td>
<td>Output</td>
<td>24-bit revocation counter to SAES peripheral</td>
</tr>
<tr>
<td>hdpl[7:0]</td>
<td>Input</td>
<td>Hide protection level stored in SBS_HDPLSR register of SBS peripheral. Protected against physical attacks. Used by embedded flash memory.</td>
</tr>
<tr>
<td>nvstate[7:0]</td>
<td>Output</td>
<td>Non-volatile security state, used by SBS peripheral</td>
</tr>
<tr>
<td>block_sec</td>
<td>Input</td>
<td>Set by device to hide rhuk and all option byte keys stored in embedded flash (read as zero). When block_sec is cleared blocked keys becomes available again, depending on HDPL value.</td>
</tr>
</tbody>
</table>

1. Only digital signals

5.3.3  FLASH architecture and integration in the system

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

The embedded flash memory offers three different interconnect interfaces:
- An AXI 64-bit slave port to access code/data
- An AHB 32-bit system slave port to access read-only and OTP area
- An AHB 32-bit configuration slave port to access register bank

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

The embedded flash memory micro-architecture is shown in *Figure 10.*
Behind the system interfaces, the embedded flash memory implements various command queues and buffers to perform flash read, write and erase operations with maximum efficiency.

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

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

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

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

The embedded flash memory is built in such a way that only one read or write operation can be executed at a time.
5.3.4 Flash memory architecture and usage

Flash memory architecture

Figure 11 shows the non-volatile memory organization in the embedded flash memory.

Figure 11. Embedded flash memory organization

The embedded flash non-volatile memory is composed of:

- A 64 Kbyte user memory block, organized as one bank divided in 8 sectors of 8 Kbytes each. This bank features flash-word rows of 128 bits + 9 bits of ECC per word. A user-defined address range benefits from a hide protection mechanism.

- A 128 Kbytes system memory block, organized as one bank divided in 16 sectors of 8 Kbytes each. The system flash memory features the same flash-word rows of 128 bits + 9 bits of ECC per word. A specific range, starting from the system boot sector, also benefit from a hide protection mechanism.

- A set of non-volatile user option byte words, loaded at reset by the embedded flash memory and accessible by the application software only through the AHB configuration register interface.

- Three sets of non-volatile option byte keys, loaded at any time by the application software through the AHB configuration register interface. All the option byte keys benefit from a hide protection mechanism.

- A 1 Kbyte one-time-programmable (OTP) area that can be written only once by the application software.

- A 512-byte read-only area written by STMicroelectronics during SoC manufacturing.
Note: The system flash memory cannot be programmed nor erased by any application. Since the OTP area has no specific protection, it must not be used to store confidential information.

The overall flash memory architecture and its corresponding access interface is summarized in Table 28.

### Table 28. Flash memory organization

<table>
<thead>
<tr>
<th>Flash memory area</th>
<th>Address range</th>
<th>Size (bytes)</th>
<th>Region name</th>
<th>Access interface</th>
<th>SSN(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>User main memory</td>
<td>0x0800 0000-0x0800 1FFF</td>
<td>8 K</td>
<td>Sector 0</td>
<td>AXI port</td>
<td>0x0</td>
</tr>
<tr>
<td></td>
<td>0x0800 2000-0x0800 3FFF</td>
<td>8 K</td>
<td>Sector 1</td>
<td></td>
<td>0x1</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>0x0800 E000-0x0800 FFFF</td>
<td>8 K</td>
<td>Sector 7</td>
<td></td>
<td>0x7</td>
</tr>
<tr>
<td>System memory</td>
<td>0x1FF0 0000-0x1FF0 1FFF</td>
<td>8 K</td>
<td>System sector 0</td>
<td>AXI port</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1FF1 E000-0x1FF1 FFFF</td>
<td>8 K</td>
<td>System sector 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Special region</td>
<td>0x08FF F000-0x08FF F3FF</td>
<td>1 K</td>
<td>OTP area</td>
<td>AHB system port</td>
<td>N/A(2)</td>
</tr>
<tr>
<td></td>
<td>0x08FF F800-0x08FF FFFF</td>
<td>512</td>
<td>Read-only area</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Option bytes</td>
<td>registers</td>
<td>-</td>
<td>User option bytes</td>
<td>Registers only</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1. SSN contains the target sector number for an erase operation. See Section 2.
2. Cannot be erased by application software

### Partition usage

In the life of the product embedded flash memory stores different kinds of data. Data availability also vary according to the hide protection level (HDPL) selected by the application at any given time. See Section 8: System configuration, boot and security (SBS) for more details.

For STM32H7R devices the typical flash usages are described on Figure 12 and below.

1. Device on a Nucleo board for s/w development, or in production warehouses.
2. Device is being provisioned with immutable root of trust firmware and root of trust keys on an untrusted OEM production line, using secure firmware install (SFI).
3. Device on the final personalization line of the customer, and as used in the final product.
1. In this state OEM_PROVD option byte is not 0xB4
2. This boot code could be protected via the HDPL area refer to Section 5.5.5: Hide protected user flash area.

For STM32H7S devices the typical flash memory usage and options are described on Figure 13 and below:

1. Device on a Nucleo board for software development, or in production warehouses.
2. Device is being provisioned with updatable root of trust firmware and root of trust keys on an untrusted OEM production line, using Secure Keys Provisioning (SKP) service. Alternatively, the device is being provisioned with immutable root of trust firmware and root of trust keys on an untrusted OEM production line, using secure firmware install (SFI).
3. Device on the final personalization line of the customer, and as used in the final product
Figure 13. Embedded flash memory usage (STM32H7S)

1. In this state OEM_PROVD option byte is not 0xB4
2. ST iRoT is selected through IROT_SELECT option byte
3. OEM iRoT is selected through IROT_SELECT option byte. When OEM iRoT executes ST iRoT is hidden.

Figure 14 shows how the embedded flash memory is used when an OEM returns a provisioned production device to an open state (for field return to ST) or to a closed state to build a product. Two decommissioning scenarios are proposed:
1. Device opening for devices with OEM iRoT
2. Device opening for devices with ST iRoT (STM32H7S only).

Note: In any device, if application knows the decommissioning key it can open the device using STM32 tools.
5.3.5 Flash hide protection schemes

*Figure 11* gives an overview of the protection mechanism supported by the embedded flash memory:

- A user flash area can be defined as user HDP area, with an hardware defined access policy based on STM32 hide protection level (HDPL).
- A system flash starting from sector zero is defined as system HDP area, also with a hardware defined access policy based on STM32 hide protection level (HDPL).
- Three sets of option byte keys are each reserved to a specific hide protection level. Some of those keys can be reserved to be used exclusively by the side channel attacks protected SAES engine, as application hardware keys (AHK).

For more information on both HDP areas, refer to *Section 5.5.4: Hide protected system flash area* and *Section 5.5.5: Hide protected user flash area*. More details on option byte keys can be found in *Section 5.4.4: Option byte key management*. 
5.3.6 Overview of FLASH operations

Read operations

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

To increase efficiency, the embedded flash memory implements the buffering of consecutive read requests.

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

Program/erase operations

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

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

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

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

Protection mechanisms

The embedded flash memory supports different protection mechanisms:

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

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

Option byte loading

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

Read operation overview

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

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

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

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

The AHB system interface operates as follows:
- Until the AHB read request has been served, the embedded flash memory stalls the AHB bus and consequently the master that issued that request.

The Read pipeline architecture is summarized in Figure 15.

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

Figure 15. FLASH read pipeline architecture
Single read sequence

The recommended simple read sequence is the following:

1. For AXI interface: Freely perform read accesses to any AXI-mapped area. For AHB interface: perform either 16-bit or 32-bit read accesses to the AHB-mapped area (byte accesses generate a bus error).

2. The embedded flash memory effectively executes the read operation from the read command queue buffer as soon as the non-volatile memory is ready and the previously requested operations have been served.

Note: When reading an OTP data that has not been previously written, a double ECC error is reported and only 1’s are returned (see Section 5.3.11 for details).

Adjusting read timing constraints

The embedded flash memory clock must be enabled and running before reading data from non-volatile memory.

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

Table 29 shows the correspondence between the number of wait states (LATENCY), the programming delay parameter (WRHIGHFREQ), the embedded flash memory clock frequency and its supply voltage ranges.

Table 29. FLASH recommended read wait states and programming delays

<table>
<thead>
<tr>
<th>Wait states per flash word read (LATENCY)</th>
<th>Read latency (cycles per flash word)</th>
<th>Programming delay (WRHIGHFREQ)</th>
<th>AXI Interface clock frequency v.s. $V_{CORE}$ range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>1</td>
<td>00</td>
<td>[0MHz ; 36MHz] [$V_{OS}$ low range 1.15 V - 1.26 V]</td>
</tr>
<tr>
<td>0x1</td>
<td>2</td>
<td>00</td>
<td>[36MHz ; 72MHz] [$V_{OS}$ low range 1.15 V - 1.26 V]</td>
</tr>
<tr>
<td>0x2</td>
<td>3</td>
<td>01</td>
<td>[72MHz ; 108MHz] [$V_{OS}$ low range 1.15 V - 1.26 V]</td>
</tr>
<tr>
<td>0x3</td>
<td>4</td>
<td>01</td>
<td>[108MHz ; 144MHz] [$V_{OS}$ low range 1.15 V - 1.26 V]</td>
</tr>
<tr>
<td>0x4</td>
<td>5</td>
<td>10</td>
<td>[144MHz ; 180MHz] [$V_{OS}$ low range 1.15 V - 1.26 V]</td>
</tr>
<tr>
<td>0x5</td>
<td>6</td>
<td>10</td>
<td>[180MHz ; 216MHz] [$V_{OS}$ low range 1.15 V - 1.26 V]</td>
</tr>
<tr>
<td>0x6</td>
<td>7</td>
<td>11</td>
<td>N/A [$V_{OS}$ high range 1.30 V - 1.40 V]</td>
</tr>
<tr>
<td>0x7</td>
<td>8</td>
<td>11</td>
<td>N/A [$V_{OS}$ high range 1.30 V - 1.40 V]</td>
</tr>
</tbody>
</table>

Adjusting the system frequency

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

When changing the AXI bus frequency, the application software must follow the below sequence in order to tune the number of wait states required to access the non-volatile memory.
To increase the embedded flash memory clock source frequency:

1. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in the FLASH_ACR register, as described in Table 29.
2. Check that the new number of wait states is taken into account by reading back the FLASH_ACR register.
3. Modify the embedded flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of the reset and clock controller (RCC).
4. Check that the new embedded flash memory clock source and/or the new AXI bus clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the AXI bus prescaler value in the RCC_CFGR register of the reset and clock controller (RCC).

To decrease the embedded flash memory clock source frequency:

1. Modify the embedded flash memory clock source and/or the AXI bus clock prescaler in the RCC_CFGR register of reset and clock controller (RCC).
2. Check that the embedded flash memory new clock source and/or the new AXI bus clock prescaler value are taken into account by reading back the embedded flash memory clock source status and/or the AXI interface prescaler value in the RCC_CFGR register of reset and clock controller (RCC).
3. If necessary, program the LATENCY and WRHIGHFREQ bits to the right value in FLASH_ACR register, as described in Table 29.
4. Check that the new number of wait states has been taken into account by reading back the FLASH_ACR register.

Read errors

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

Single error correction and double error detection are performed for each read operation. In both cases the embedded flash memory reports read errors as described in Section 5.7.6: Error correction code error (SNECCERRF/DBECCERRF).

Read interrupts

See Section 5.8: FLASH interrupts for details.

5.3.8 FLASH program operations

Program operation overview

The virgin state of each non-volatile memory bitcell is 1. The embedded flash memory supports programming operations that can change any memory bitcell to 0. However these operations do not support the return of a bit to its virgin state. In this case an erase operation of the entire sector is required.
Program operation consists in issuing write commands. For write accesses issued by the AXI interface, since a 9-bit ECC code is associated to each 128-bit data flash word, the embedded flash memory must always perform write operations to non-volatile memory with a 128-bit word granularity.

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

System flash memory cannot be written by the application software.

It is not recommended to overwrite a flash word that is not virgin. The result may lead to an inconsistent ECC code that is systematically reported by the embedded flash memory, as described in Section 5.7.6: Error correction code error (SNECCERRF/DBECCERRF).

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

**Note:** The OTP area is typically write-protected on the final product, as described in Section 5.3.11: FLASH one-time programmable area.

Erase operations to the OTP area are not supported.

The AXI interface write channel operates as follows:

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

The AHB system interface operates as follows:

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

The write pipeline architecture is described in Figure 16.

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

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

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

- If a write protection violation is detected, the write operation is canceled and write protection error (WRPERRF) is raised in FLASH_ISR register.
- If the write operation is valid, the 9-bit ECC code is concatenated to the 128 bits of data and the write to non-volatile memory is effectively executed.

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

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

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

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

Monitoring ongoing write operations

The application software can use three status flags located in FLASH_SR in order to monitor ongoing write operations.

- **BUSY**: this bit indicates in real-time that an effective write, erase or option byte change operation is ongoing to the non-volatile memory. It is possible to know what type of
operation is being executed polling the bits IS_PROGRAM, IS_ERASE and IS_OPTCHANGE in the same register.

- **QW**: this bit indicates that a write, erase or option byte change operation is pending in the write queue or command queue buffer. It remains high until the write operation is complete. It supersedes the BUSY status bit.

- **WBNE**: this bit indicates that the embedded flash memory is waiting for new data to complete the 128-bit write buffer. In this state the write buffer is not empty. It is reset as soon as the application software fills the write buffer, force-writes the operation using FW bit in FLASH_CR, or disables all write operations.

**Enabling write operations**

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

Before programming an option byte word or an option byte key, the application software must make sure that PG_OPT bit is set in FLASH_OPTCR. If it is not set an unlock sequence must be used once (see Section 5.5.1: FLASH configuration protection). For more information on option byte word (resp. option byte key) modification, refer to Section 5.4.3 (or Section 5.4.4 respectively).

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

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

**Single write sequence**

The recommended single write sequence is the following:

1. Unlock the FLASH_CR register, as described in Section 5.5.1: FLASH configuration protection (only if register is not already unlocked).
2. Enable write operations by setting the PG bit in the FLASH_CR register.
3. Check the protection of the targeted memory area.
4. Write one flash-word corresponding to 16-byte data starting at 16-byte aligned address.
5. Check that QW has been raised and wait until it is cleared to 0.

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

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

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

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

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

The user must therefore trim the optimal programming delay through the WRHIGHFREQ parameter in the FLASH_ACR register. Refer to Table 29 in Section 5.3.7: FLASH read operations for the recommended programming delay depending on the embedded flash memory clock frequency.

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

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

Programming errors

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

Programming interrupts

See Section 5.8: FLASH interrupts for details.

5.3.9 FLASH erase operations

Erase operation overview

The embedded flash memory can perform erase operations (if allowed) on 8-Kbyte user sectors or on the whole user flash memory.

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

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

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

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

Erase and security

A user sector can be erased only if it does not contain hide protected or write-protected data (see Section 5.5: FLASH protection mechanisms for details). In other words, if the application software attempts to erase a user sector with at least one flash word that is protected, the sector erase operation is aborted and the WRPERRF flag is raised in the FLASH_ISR register, as described in Section 5.7.2.
The only method to erase HDP protected sectors is to perform a regression, triggering a
NVSTATE change from CLOSE to OPEN. See Section 5.5.3 for details.

**Enabling erase operations**

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

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

Similar constraints apply to bank erase requests.

**Flash sector erase sequence**

To erase a 8-Kbyte user sector, proceed as follows:

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

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

**Flash bank erase sequence**

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

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

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

If a sector erase is requested simultaneously to the bank erase (SER bit set), the bank
erase operation supersedes the sector erase operation.
5.3.10 Flash memory error protections

Error correction codes (ECC)

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

Per 128-bit system and user flash word the embedded flash memory uses 9 ECC bits. During each read operation from a 128-bit flash word, FLASH retrieves the 9-bit ECC information, computes the ECC of the flash word, and compares the result with the reference value. If they do not match, the corresponding ECC error is raised as described in Section 5.7.6: Error correction code error (SNECCERRF/DBECCERRF). During each program operation, a 9-bit ECC code is associated to each 128-bit data flash word, and the resulting 137-bit flash word information is written in non-volatile memory.

For read-only and OTP areas, the embedded flash memory uses a stronger 6 ECC bits per 16-bit word. For option byte words and keys a double ECC scheme is used. For more information on read-only and OTP refer to Section 5.3.11 and Section 5.3.12. For more information on option byte words and keys refer to Section 5.4.

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

Cyclic redundancy codes (CRC)

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

The area processed by the CRC module can be defined either by sectors or by start/end addresses.

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

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

\[ x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \]

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

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

1. Unlock FLASH_CR register, if not already unlocked.
2. Enable the CRC feature by setting the CRC_EN bit in FLASH_CR.
3. Program the desired data size in the CRC_BURST field of FLASH_CRCRR.
4. Define the user flash memory area on which the CRC has to be computed. Three solutions are possible:
   - Select all user flash sectors by setting the ALL_SECT bit in FLASH_CRCRR
   - Define the area start and end addresses by programing FLASH_CRCADDR and FLASH_CRCADDR, respectively,
   - Select the targeted sectors by setting the CRC_BY_SECT bit in FLASH_CRCRR.

Also program consecutively the target sector numbers using the CRC_SECT field
of the FLASH_CRCCR register. Set ADD_SECT bit after each CRC_SECT programming.

5. When step 4 is completed start the CRC operation by setting the START_CRC bit in FLASH_CRCCR.

6. Wait until the CRC_BUSY flag is reset in FLASH_SR.

7. Retrieve the CRC result in FLASH_CRCDATAR.

**Note:** The application should avoid running a CRC on HDP user flash memory area when the hide protection level prevents access to it. Indeed, doing so may alter the expected CRC value. A special error flag defined in Section 5.7.8: CRC read error (CRCRDERRF) can be used to detect such a case.

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

### 5.3.11 FLASH one-time programmable area

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

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

OTP data can be accessed through the AHB system port. They are organized as 16 blocks of 32 OTP words, as shown in Table 30. An entire OTP block can be protected (locked) from write accesses by setting the corresponding OTPL bit in the FLASH_OTPLSRP register, as shown. There is no special read protection mechanism on the OTP area.

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

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

<table>
<thead>
<tr>
<th>OTP block</th>
<th>AHB address</th>
<th>AHB word [31:16]</th>
<th>AHB word [15:0]</th>
<th>Lock bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 0</td>
<td>0x08FF F000 OTP001 OTP000</td>
<td>OTPL[0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F004 OTP003 OTP002</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F03C OTP031 OTP030</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1</td>
<td>0x08FF F040 OTP033 OTP032</td>
<td>OTPL[1]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F044 OTP035 OTP034</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F07C OTP063 OTP062</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 2</td>
<td>0x08FF F080 OTP065 OTP064</td>
<td>OTPL[2]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F084 OTP067 OTP066</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F0BC OTP127 OTP126</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
OTP error protection

OTP data are organized as 16 blocks of 32 OTP words, as shown on Table 30. Each 16-bit half-word is protected by 6 bits of ECC. Hence application must avoid overwriting a 16-bit half-word that has already been programmed, as it is leading to systematic ECC error. Similarly, do not write twice an 16-bit OTP word, as it could lead to systematic ECC error.

When reading OTP data with a single error corrected or a double error detected, the embedded flash memory reports the corresponding error, as described in Section 5.7.6: Error correction code error (SNECCERRF/DBECCERRF).

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

OTP write sequence

Follow the sequence below to write an OTP word:

1. Check the protection status of the target OTP word (see Table 30). The corresponding OTPLi bit must be cleared in FLASH_OTPLSR register.
2. Unlock FLASH_CR if the register is not already unlocked, and then set PG_OTP bit in the FLASH_CR register.
3. Write two OTP words (32 bits) corresponding to the 4-byte aligned address shown in Table 30. Alternatively, the application software can program separately the 16-bit MSB or 16-bit LSB. In this case the first 16-bit write operation starts immediately without waiting for the second one.
4. Check that the QW bit in FLASH_SR has been set and wait until it is cleared.
5. Optionally, lock the OTP block using OTPLi bit in FLASH_OTPLSRP in order to prevent further data changes. PG_OTP bit in the FLASH_CR register can also be cleared if needed.

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

No error sequence and no inconsistency error are generated during OTP write operations. Writing OTP data at byte level is not supported and generates a bus error.

Table 30. Flash memory OTP organization (continued)

<table>
<thead>
<tr>
<th>OTP block</th>
<th>AHB address</th>
<th>AHB word [31:16]</th>
<th>[15:0]</th>
<th>Lock bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 15</td>
<td>0x08FF F3C0</td>
<td>OTP481</td>
<td>OTP480</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x08FF F3C4</td>
<td>OTP483</td>
<td>OTP482</td>
<td>OTPL[15]</td>
</tr>
<tr>
<td></td>
<td>0x08FF F3FC</td>
<td>OTP511</td>
<td>OTP510</td>
<td></td>
</tr>
</tbody>
</table>

...
5.3.12 FLASH read-only area

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

Read-only area can be accessed through the AHB system port. It is organized as shown in Table 31. This information is provisioned by STMicroelectronics during the device manufacturing.

<table>
<thead>
<tr>
<th>AHB address (read-only)</th>
<th>Read-only data name and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x08FF F800</td>
<td>U_ID[31:0]</td>
</tr>
<tr>
<td>0x08FF F804</td>
<td>U_ID[63:32]</td>
</tr>
<tr>
<td>0x08FF F808</td>
<td>U_ID[95:64]</td>
</tr>
<tr>
<td>0x08FF F80C</td>
<td>Flash memory size[15:0]</td>
</tr>
<tr>
<td>0x08FF F810 to 0x08FF F9FF</td>
<td>Reserved information</td>
</tr>
</tbody>
</table>

Read-only area error protection

Read-only area is protected by a robust ECC scheme. When reading a 32-bit word in this area with a single error corrected or a double error detected, the embedded flash memory reports the corresponding error, as described in Section 5.7.6: Error correction code error (SNECCERRF/DBECCERRF).

5.3.13 FLASH reset and clocks

Reset management

Following a power-on reset embedded flash selects two security states, as shown on Figure 17. For more details on those states, refer to Section 5.5.3.

Figure 17. FLASH stateful initialization

Embedded flash memory stateful initialization is based on the overloading of option bytes default values by non-volatile values automatically loaded when the reset signal rises.
During this loading sequence, other parts of the device remains under reset and the embedded flash memory is not accessible from its interfaces.

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

- All registers, except for option byte registers, are cleared, including read and write latencies. Option byte register list is found in Section 5.4.5.
- Most control registers are automatically protected against write operations, as described in Section 5.5.1: FLASH configuration protection.

It is important to note that the contents of the flash memory (except option bytes) are not guaranteed if a power-on-reset occurs during a flash memory write or erase operation. For option bytes a valid content is always guaranteed, as embedded flash uses the old option byte values when an option byte modification is interrupted by a reset. In this case a new option byte change request must be issued to modify those option bytes.

**Clock management**

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

### 5.4 FLASH option bytes

#### 5.4.1 About option bytes

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

This section explains:

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

#### 5.4.2 Option byte loading

When the device is first powered, the embedded flash memory automatically loads all the option byte words, and few selected option byte keys. During the option byte loading sequence, the device remains under reset and the embedded flash memory cannot be accessed.

When an ECC issue is detected during this option byte loading sequence OBLERRF flag is raised, as described in Section 5.7.12.
5.4.3 Option byte words modification

Changing option byte words

A option byte word change operation can modify the configuration and the protection settings saved in the non-volatile option byte area, if allowed.

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

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

When option byte word register “SRP” is written the embedded flash memory checks if at least one option byte needs to be programmed by comparing the current values in corresponding “SR” register. If a change is detected, if associated change conditions are met (see Changing security option bytes) and if PG_OPT bit is set in FLASH_OPTCR, the embedded flash memory launches the option byte modification in its non-volatile memory and update the “SR” register.

If one of the condition described in Changing security option bytes is not respected, the embedded flash memory sets the OPTERRF flag in the FLASH_OPTISR register and aborts the option byte change operation.

Unlocking the option byte modification

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

Option byte modification sequence

To modify an option byte word, follow the sequence below:
1. Unlock FLASH_OPTCR register as described in Section 5.5.1: FLASH configuration protection (only if the register is not already unlocked).
2. Enable the write operations by setting the PG_OPT bit in FLASH_OPTCR.
3. Write the desired new option byte values in the corresponding option byte word register (FLASH_xSRP).
4. Wait until the QW bit is cleared in FLASH_SR register. Once cleared corresponding FLASH_xSR register has been updated, if security allows it (see next section).

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

On top of OPTLOCK bit, there is a second level of protection for security-sensitive option byte words. Note that the option byte words defined in Section 5.4.5 do not have any special protection.

Option byte words with specific protections are described hereafter.

- **Sector write protection (WRPS)**
  Those user options managing group of sector write protection can be changed using FLASH_WRPSRP register, if application is RSS or the immutable root of trust.

- **Non-volatile state (NVSTATE)**
  A detailed description of NVSTATE option bits is given in Section 5.5.3. The following rules must be respected to modify NVSTATE using FLASH_NVSRP register.
  - Writes to FLASH_NVSRP register are ignored if HDPL is different from 0 or 1 in SBS_HDPLSR register.
  - NVSTATE option change is triggered only after the RAM erase signal has confirmed that Backup RAM and PKA RAM have been successfully erased.
  - When NVSTATE=OPEN, option byte change to CLOSE is allowed. Writing any other value than 0x51 triggers an OPTERRF flag, without option change.
  - When NVSTATE=CLOSE, option byte change to OPEN is allowed, triggering a global erase explained in Section 5.5.3. Writing any other value than 0xB4 triggers an OPTERRF flag, without option change.

- **OEM provisioned (OEM_PROVD)**
  As defined in Table 36: Boot level and HDP area protections (non STiRoT case), OEM_PROVD defines the protection of the HDP area. OEM_PROVD can only be changed by RSS in FLASH_ROTSPR.

- **iRoT selection (IROT_SELECT)**
  As defined in Table 37: Boot level and HDP area protections (STiRoT case), IROT_SELECT defines where the immutable root of trust code is stored, for STM32H7S devices only. IROT_SELECT can only be changed when the device is opened by RSS in FLASH_ROTSPR.

- **Debug authentication method (DBG_AUTH)**
  DBG_AUTH defines the method used to open up the device debug. It has no hardware effect in embedded flash memory.
  DBG_AUTH can only be changed by RSS in FLASH_ROTSPR.

- **Secure storage counter (EPOCH)**
  This 24-bit non-volatile value is directly send by embedded flash to the SAES peripheral. As soon as this value changes all current keys encrypted with a derived hardware unique key (DHUK) are lost. For more details on the DHUK, refer to SAES chapter. EPOCH can only be changed by RSS.

- **Hide protection area size (HDP_AREA_START and HDP_AREA_END)**
  These user options configure the size of the user flash area that can be accessed while HDPL equals to the value defined in Section 5.5.5. Writes to FLASH_HDPSRP register are ignored if HDPL is different from 0 or 1 in the SBS_HDPLSR register.

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

Hide protection area end address in system flash is fixed by STMicroelectronics.
5.4.4 Option byte key management

Option byte key programming

Following sequence is required to program an option byte key in embedded flash memory:

1. Unlock FLASH_OPTCR register as described in Section 5.5.1: FLASH configuration protection (only if the register is not already unlocked).

2. Enable the write operation by setting the PG_OPT bit in FLASH_OPTCR, and make sure the KTEF and KVEF bits are cleared in FLASH_OPTISR.

3. Write the key value in the FLASH_OBKDRx registers. Key[31:0] goes in OBKDR0[31:0], Key[63:32] to OBKDR1[31:0], Key[95:64] to OBKDR2[31:0], Key[127:96] to OBKDR3[31:0], Key[159:128] to OBKDR4[31:0], Key[191:160] to OBKDR5[31:0], Key[223:192] to OBKDR6[31:0] and Key[255:224] to OBKDR7[31:0].

4. Fill the following information to FLASH_OBKCR:
   - OBKINDEX[4:0]: index of the key for this hide protection level
   - OBKSIZE[1:0]: size of the key (see bitfield description)
   - NEXTKL[1:0]: when HDPL=1 in SBS_HDPLSR register RoT application can set those bit to 01 to provision a key for the next HDPL level (2).
   - Set both KEYSTART and KEYPROG bits to start the programming sequence.

5. Read back the key to verify its value (recommended). See Option byte key loading.

Note: When HDPL value in SBS peripheral changes FLASH_OBKDR register is cleared to 0x0.

For a given hide protection level, if authorized the application writes to an OBKINDEX that has already been successfully used before the old key is overwritten with the new value.

Like for any option change, if an error occurs OPTERRF is set in FLASH_OPTISR. It must be cleared, like KTEF and KVEF, before initiating another option byte key programming (otherwise the setting of KEYSTART bit is ignored).

All key error flags (KVEF, KTEF) must be cleared before application reads or write an option byte key.

Note: Option byte key integrity could fail when a reset is issued while programming it. When this happens the previously written key is kept, and no key is returned if it was the first programming. The OBLERRF error flag is also raised.

Option byte key loading

If OBKINDEX[4:0] is or greater than or equal to 0x8, the following sequence is required to read an option byte key stored in embedded flash memory:

1. Fill the following information to FLASH_OBKCR:
   - OBKINDEX[4:0]: index of the key for this hide protection level. Size of the key is automatically recovered from the flash memory.
   - NEXTKL[1:0]: when HDPL=1 in SBS_HDPLSR register RoT application can set those bit to 01 to read a key provisioned for the next HDPL level (2).
   - Set KEYSTART bit with KEYPROG bit cleared to start the read sequence.

2. Wait until QW bit is cleared in FLASH_SR register.

3. Read the value of the key using FLASH_OBKDRx registers

If OBKINDEX[4:0] is less than 0x8, the following sequence is required to load to the SAES peripheral an option byte key stored in embedded flash.
1. In the SAES peripheral, set the correct KEYSIZE and write KEYSEL[2:0] =011 or 101 in the SAES_CR register.

2. Perform the same sequence as a normal option byte key read, with the exception of reading back the FLASH_OBKDRx registers. In this case, key data are automatically transferred from embedded flash memory to the SAES key registers, while the BUSY bit is set in the SAES_SR register.

3. Once the transfer is completed the BUSY bit is cleared and the KEYVALID bit is set in the SAES_SR register. If KEYVALID is not set when the BUSY bit is cleared, or if a key error flag (KEIF) is set in SAES, this means that an unexpected event occurred during the transfer. In this unlikely event the IPRST bit must be set then cleared in the SAES_CR register, then sequence can be restarted from Step 1 above.

During option byte key loading two errors can happen:
- Key valid error (see Section 5.7.10)
- Key transfer error (see Section 5.7.11)

All key error flags (KVEF, KTEF) must be cleared before application reads or writes an option byte key.

Note: When the HDPL value in the SBS peripheral changes, the FLASH_OBKDR register is cleared to 0x0.

5.4.5 Option byte user words overview

Table 32 lists all the user option bytes managed through the embedded flash memory registers, as well as their default values before the first option byte change (default factory value). They are configured by the end-user depending on the application requirements. Some option bytes might have been initialized by STMicroelectronics during manufacturing stage.

Table 32. Option byte user words organization

<table>
<thead>
<tr>
<th>Register</th>
<th>Bitfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_OBW1SR[31:16]</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 0</td>
</tr>
<tr>
<td>FLASH_OBW1SR[15:0]</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0</td>
</tr>
</tbody>
</table>
Table 32. Option byte user words organization (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Bitfield</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_OBW2SR[15:0]</td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 1 0 1 1 1 0 1 1 1 0x000</td>
</tr>
<tr>
<td>FLASH_HDPSR[31:16]</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td>HDP_AREA_END</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0x00</td>
</tr>
<tr>
<td>FLASH_HDPSR[15:0]</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>HDP_AREA_START</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0x0ffe</td>
</tr>
<tr>
<td>FLASH_WRPSR[15:0]</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>FLASH_OTPLSR[15:0]</td>
<td>OTPL[51:48]</td>
</tr>
<tr>
<td></td>
<td>OTPL[48:449]</td>
</tr>
<tr>
<td></td>
<td>OTPL[63:32]</td>
</tr>
<tr>
<td></td>
<td>OTPL[31:0]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>FLASH_NVSR[15:0]</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>NVSTATE</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>0 0 0 0 0 0 0 1 0 1 1 0 1 0 0</td>
</tr>
<tr>
<td>FLASH_ROTMRP[15:0]</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td>IROT_SELECT</td>
<td></td>
</tr>
<tr>
<td>Default factory value</td>
<td>1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Embedded flash memory (FLASH)
5.4.6 Description of user option byte word

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

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

*Note:* If the hardware watchdog “control selection” feature is enabled (set to 0), the watchdog is automatically enabled at power-on, thus generating a reset unless the watchdog key register is written to or the down-counter is reloaded before the end-of-count is reached. Depending on the configuration of IWDG_STOP and IWDG_STBY options, the IWDG can continue counting (1) or not (0) when the device is in Stop or Standby mode respectively. When the IWDG is kept running during Stop or Standby mode, it can wake up the device from these modes.

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

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

- **Device options**
  - VDDIO_HSLV: enables the configuration of pads below 2.5 V for VDDIO power rail if set to 1.
  - XSPI1_HSLV: enable I/O XSPIM_P1 high-speed option if set to 1.
  - XSPI2_HSLV: enable I/O XSPIM_P2 high-speed option if set to 1.

When STMicroelectronics delivers the device, the values programmed in the general-purpose option bytes are the following:

- **Watchdog management**
  - IWDG reset active in Standby and Stop modes (option value = 0x1)
  - IWDG not automatically enabled at power-on (option byte value = 0x1)

- **Reset management:**
  - BOR: brownout level option (reset level) equals 2.1 V (option byte value = 0x0). A reset is not generated when the device enters Standby or Stop low-power mode (option byte value = 0x1)

- The speed of the device I/Os are optimized when device voltage is low (VDDIO_HSLV= XSPI1_HSLV= XSPI2_HSLV= 1)

Refer to *[Section 5.9: FLASH registers]* for details.
5.4.7 Description of security option bytes

The option bytes that are use by root of trust application to enhance security are listed below:

- NVSTATE[7:0]: Non-volatile state (see Section 5.5.3 for details).
- IROT_SELECT[7:0]: Selection of ST iRoT (STM32H7S only).
- WRPS[7:0]: Write protection option of user flash sectors. It is active low. Refer to Section 5.9.26 for details.
- OTPL[15:0]: Write protection option of OTP blocks. It is active high. Refer to Section 5.3.11 for details.
- HDP_AREA: Hide protection user flash area definitions. Refer to Section 5.5.5.
  - HDP_AREA_START (respectively HDP_AREA_END) contains the first (respectively last) 256-byte block of the HDP area.
Usage of above options are summarized in Table 33.

Table 33. STM32H7Rx/7Sx device lifecycle table

<table>
<thead>
<tr>
<th>Product state</th>
<th>NVSTATE</th>
<th>OEM_PROVD</th>
<th>DBG_AUTH</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open</td>
<td>OPEN (0xB4)</td>
<td>x</td>
<td>x</td>
<td>Device delivered by ST. User flash memory usage is open, and debug access is enabled.</td>
</tr>
<tr>
<td>Provisioning</td>
<td>CLOSE (0x51)</td>
<td>Not 0xB4</td>
<td>Not set (0x00)</td>
<td>Device on the production line. User flash usage is restricted, and debug access is restricted.</td>
</tr>
<tr>
<td>Locked</td>
<td></td>
<td></td>
<td>Locked (0xB4)</td>
<td>Final product with root of trust and no debug. Transition to other state or policy change (including debug) is not permitted.</td>
</tr>
<tr>
<td>Closed(1)</td>
<td></td>
<td></td>
<td>ECDSA sign (0x51)</td>
<td>Final product, with root of trust and authenticated debug activated. Regression is possible.</td>
</tr>
</tbody>
</table>

1. A “closed” product with incorrect or non-defined debug certificate is equivalent to a “locked” product.

When STMicroelectronics delivers the device, the values programmed in the security option bytes are the following:

- OPEN device (NVSTATE value = 0xB4)
- iRoT selection (only on H7S part)
  - OEM iRoT (IROT_SELECT value = 0x6A)
  - ST iRoT (IROT_SELECT value = 0xB4)
- Swap is locked deactivated, with all user flash sector are allocated to bank A.
- HDP area protection disabled (start addresses higher than end addresses)
- Write protection disabled (all option byte bits set to 1)
- All OTP blocks are writable, as they are virgin (all option byte bits set to 0)

Refer to Section 5.9: FLASH registers for details.
5.5 **FLASH protection mechanisms**

Since sensitive information are stored in the flash memory, it is important to protect it against unwanted operations such as reading confidential areas, illegal programming of immutable sectors, or malicious flash memory erasing.

For this purpose FLASH implements the following protection mechanisms:

- Configuration protection
- User flash write protection
- Device non-volatile security life cycle and application temporal isolation management
- OTP locking

This section provides a detailed description of all these security mechanisms.

5.5.1 **FLASH configuration protection**

The embedded flash memory uses hardware mechanisms to protect the following assets against unwanted or spurious modifications (e.g. software bugs):

- Option bytes change
- Write operations
- Erase commands
- Interrupt masking

More specifically, write operations to embedded flash memory control registers (FLASH_CR and FLASH_OPTCR) are not allowed after reset.

The following sequence must be used to unlock FLASH_CR register:

1. Program KEY1 to 0x45670123 in FLASH_KEYR key register.
2. Program KEY2 to 0xCDEF89AB in FLASH_KEYR key register.
3. LOCK bit is now cleared and FLASH_CR is unlocked.

The following sequence must be used to unlock FLASH_OPTCR register:

1. Program OPTKEY1 to 0x08192A3B in FLASH_OPTKEYR option key register.
2. Program OPTKEY2 to 0x4C5D6E7F in FLASH_OPTKEYR option key register.
3. OPTLOCK bit is now cleared and FLASH_OPTCR register is unlocked.

Any wrong sequence locks up the corresponding register/bit until the next system reset, and generates a bus error.

The above registers can be locked again by software by setting the LOCK bit in the corresponding control register.

The FLASH_CR register remains locked and a bus error is generated when the following operations are executed:

- Programming a third key value
- Writing to a different register than FLASH_KEYR before FLASH_CR has been completely unlocked (KEY1 programmed but KEY2 not yet programmed)
- Writing less than 32 bits to KEY1 or KEY2.
Similarly the FLASH_OPTCR register remains locked and a bus error is generated when the following operations are executed:

- Programming a third key value
- Writing to a different register before FLASH_OPTCR has been completely unlocked (OPTKEY1 programmed but OPTKEY2 not yet programmed)
- Writing less than 32 bits to OPTKEY1 or OPTKEY2.

The embedded flash memory configuration registers protection is summarized in Table 34. Registers not present in this table are not protected by any key.

### Table 34. Flash interface register protection summary

<table>
<thead>
<tr>
<th>Register name</th>
<th>Unlocking register</th>
<th>HDP level(1)</th>
<th>Protected asset</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLASH_CR</td>
<td>FLASH_KEYR</td>
<td>Any</td>
<td>Flash write operations, erase commands</td>
</tr>
<tr>
<td>FLASH_IER</td>
<td>FLASH_CR</td>
<td></td>
<td>Interrupt generation masking sources</td>
</tr>
<tr>
<td>FLASH_CRCRR</td>
<td>FLASH_CR</td>
<td></td>
<td>Flash bank CRC operations</td>
</tr>
<tr>
<td>FLASH_OPTCR</td>
<td>FLASH_OPTKEYR</td>
<td></td>
<td>Option bytes changes</td>
</tr>
<tr>
<td>FLASH_OBKCR</td>
<td>FLASH_OPTKEYR</td>
<td></td>
<td>Option byte key change</td>
</tr>
<tr>
<td>FLASH_NVSRP</td>
<td>FLASH_OPTCR</td>
<td>0 or 1</td>
<td>Non-volatile state change</td>
</tr>
<tr>
<td>FLASH_HDPSRP</td>
<td>FLASH_OPTCR</td>
<td>0</td>
<td>Hide protect area change</td>
</tr>
<tr>
<td>FLASH_EPOCHSRP</td>
<td>FLASH_OPTCR</td>
<td>0</td>
<td>Epoch change</td>
</tr>
<tr>
<td>FLASH_OBWSRxP</td>
<td>FLASH_OPTCR</td>
<td>Any</td>
<td>User option byte word x change</td>
</tr>
<tr>
<td>FLASH_WRPSRP</td>
<td>FLASH_OPTCR</td>
<td></td>
<td>User flash write protection change</td>
</tr>
<tr>
<td>FLASH_OTPLSRP</td>
<td>FLASH_OPTCR(2)</td>
<td>Any</td>
<td>OTP write protection</td>
</tr>
</tbody>
</table>

1. Required HDP level in the device
2. Once set, the OTP Block Lock bits cannot be reset by the application.

### 5.5.2 Write protection

The purpose of embedded flash memory write protection is to prevent unwanted modifications to embedded non-volatile code and/or data.

Any 8 Kbyte sector can be independently write-protected or unprotected by clearing/setting the corresponding WRPS bit in the FLASH_WRPSR register.

A write-protected sector can neither be erased nor programmed. As a result, a full bank erase cannot be performed if one sector is write-protected, unless a NVSTATE transition to OPEN is triggered.

The embedded flash memory write-protection user option bits can be modified when HDPL=0 or 1 in SBS_HDPLSR register, using FLASH_WRPSRP.

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

Write protection errors are documented in Section 5.7: FLASH error management.
5.5.3 Life cycle management

Introduction

Figure 18 shows how embedded flash manages the non-volatile security life cycle of the device.

Figure 18. Life cycle management scheme

Table 35. Flash security lifecycle definition

<table>
<thead>
<tr>
<th>NVSTATE</th>
<th>Boot address(^{(1)})</th>
<th>Debug level (^{(2)})</th>
<th>Regression to</th>
<th>Root HUK</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>0x0800 0000 or Bootloader</td>
<td>Unconstrained</td>
<td>n/a</td>
<td>0x0</td>
</tr>
<tr>
<td>CLOSE</td>
<td>0x1FF0 0000</td>
<td>Constrained (^{(3)})</td>
<td>OPEN</td>
<td>OB Key (ST)</td>
</tr>
</tbody>
</table>

1. Enforced outside flash.
2. Includes STMicroelectronics engineering test modes. Those are disabled in CLOSE state, by hardware.
3. User can attempt to unlock debug features, if allowed by DEBUG_AUTH option bits.

Flash opening

When requesting a change of NVSTATE to OPEN, writing 0xB4 to FLASH_NVSRP register, flash performs an automatic erase of the following information:

- The whole user flash (write protected or not)
- The write protection data in FLASH_WRPSR.
- The option byte register defining the hide protection area is reset to their default manufacturing values (disabled).
- The option byte keys, excluding hide protection level 0 keys
- Root of trust option byte registers: FLASH_ROTSR and FIXEDSR

Note: The FLASH_EPOCHSR setting is kept when opening the flash memory

NVSTATE option change is triggered only after the RAM erase signal has confirmed that Backup RAM and PKA RAM have been successfully erased.
The system flash memory and the read-only region are not affected by flash opening and remain unchanged.

**Flash closing**

Writing 0x51 to the FLASH_NVSRP register triggers a change of NVSTATE from OPEN to CLOSE, confirmed when NVSTATE=0x51 in FLASH_NVSR register.

### 5.5.4 Hide protected system flash area

System flash sectors included between 0x1FF0 0000 and SEC_AREA_END addresses can be used to store STMicroelectronics secure firmware install (SFI) code, or the STM32 immutable root of trust code and data, with the associated encrypted keys stored in option byte keys.

Embedded flash system area can only be accessed by Cortex M7 in read or execute. The hide protected area can only be accessed when NVSTATE is CLOSE and when hide protection level in SBS_HDPLSR is 0 or 1 for ST iRoT products (default), or just 0 for OEM iROT products.

In all other cases one of the following events is triggered:

- Data read transactions return zero. Data write transactions are ignored. No bus error is generated but an error flag is raised (RDSERR for read, WRPERR for write).
- Read instruction transactions generate a bus error and the RDSERR error flag is raised.

Embedded flash system area is immutable, that is, it cannot be erased by any application code.

*Note:* When FLASH transition from CLOSE to OPEN state, embedded flash system area is kept.

### 5.5.5 Hide protected user flash area

**Overview**

User flash sectors included between HDP_AREA_START and HDP_AREA_END addresses are used to store the root of trust code and data, with the associated keys stored in embedded flash option byte keys.

Embedded flash hide protected (HDP) user area can only be accessed:

- by Cortex M7 in read, write or execute
- according to the conditions defined in following tables. *Table 36* concerns STM32H7R parts, or STM32H7S parts with IROT_SELECT different from 0xB4. STM32H7S parts with ST iRoT (IROT_SELECT=0xB4) are described in *Table 37*. 
Table 36. Boot level and HDP area protections (non STiRoT case)

<table>
<thead>
<tr>
<th>Product state</th>
<th>FLASH user options(1)</th>
<th>HDP level(2) to activate protection</th>
<th>Boot address (HDPL)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NV STATE</td>
<td>OEM PROVD</td>
<td>HDP area in system flash</td>
<td>HDP area in user flash</td>
</tr>
<tr>
<td>Open</td>
<td>OPEN</td>
<td>not 0xB4</td>
<td>&gt;0</td>
<td>&gt;1</td>
</tr>
<tr>
<td>Provisioning</td>
<td>CLOSE</td>
<td></td>
<td>&gt;0</td>
<td>&gt;0</td>
</tr>
<tr>
<td>Closed or Locked</td>
<td>0xB4</td>
<td></td>
<td>&gt;1</td>
<td></td>
</tr>
</tbody>
</table>

1. In FLASH_NVSR and FLASH_ROTTSR registers
2. Level for which hide protection is activated.

Table 37. Boot level and HDP area protections (STiRoT case)

<table>
<thead>
<tr>
<th>Product state</th>
<th>FLASH user options(1)</th>
<th>HDP level(2) to activate protection</th>
<th>Boot address (HDPL)</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NV STATE</td>
<td>OEM PROVD</td>
<td>HDP area in system flash</td>
<td>HDP area in user flash</td>
</tr>
<tr>
<td>Open</td>
<td>OPEN</td>
<td>not 0xB4</td>
<td>&gt;0</td>
<td>&gt;2</td>
</tr>
<tr>
<td>Provisioning</td>
<td>CLOSE</td>
<td></td>
<td>&gt;1</td>
<td>&gt;1</td>
</tr>
<tr>
<td>Closed or Locked</td>
<td>0xB4</td>
<td></td>
<td>&gt;2</td>
<td></td>
</tr>
</tbody>
</table>

1. In FLASH_NVSR and FLASH_ROTTSR registers
2. Level for which hide protection is activated.

For more details on HDPL level following a reset, refer to Section 8: System configuration, boot and security (SBS).

Note: IROT_SELECT is stored in the FLASH_ROTTSR register.
Hide protection area in user flash programming

HDP user flash area is programmed in FLASH_HDPSRP by setting the HDP_AREA_START and HDP_AREA_END so that the END address is strictly higher than the START address.

HDP_AREA_START and HDP_AREA_END are defined with a granularity of 256 bytes. This means that the actual HDP user flash area size (in bytes) is defined by:

\[(HDP\_AREA\_END - HDP\_AREA\_START + 1) \times 256\]

As an example, to set the HDP user flash area on the first 8 Kbytes (that is, from address 0x0800 0000 to address 0x0800 1FFF, both included), the embedded flash memory must be configured as follows:

\[
\begin{align*}
HDP\_AREA\_START[15:0] &= 0x0 \\
HDP\_AREA\_END[15:0] &= 0x001F
\end{align*}
\]

The HDP user flash area size defined above is equal to:

\[(HDP\_AREA\_END - HDP\_AREA\_START + 1) \times 256 = 32 \times 256 \text{ bytes} = 8 \text{ Kbytes}.\]

The minimum HDP user flash area that can be set is 32 flash words (or 512 bytes). The maximum area is the whole user flash memory bank, configured by setting HDP\_AREA\_START= HDP\_AREA\_END.

Note: It is recommended to align the HDP user area size with 8 Kbytes flash sector granularity to always be able to erase non-HDP code and data.

It is possible to disable HDP protections by setting HDP\_AREA\_END lower than HDP\_AREA\_START.

Hide protection area in user flash properties

- Embedded flash hide protected user area can only be accessed in the conditions described in Overview. In all other cases one of the following events is triggered:
  - Data read transactions return zero. Data write transactions are ignored. No bus error is generated but an error flag is raised (RDSERR for read, WRPERR for write).
  - Read instruction transactions generate a bus error and the RDSERR error flag is raised.
- HDP user flash area is erase-protected. As a result:
  - No erase operations to a sector located in this area are possible (including the sector containing the area end address), unless HDPL=0 or 1.
  - In HDPL=2 or 3 the only possibility to fully erase user flash bank is to perform a device opening sequence (see Section 5.5.3).
- Rules to change the HDP user flash area are detailed in Changing option byte words in Section 5.4.3.
- When FLASH transition from CLOSE to OPEN state, the whole user flash (hide protected or not) is automatically erased, with the HDP area disabled (see Section 5.5.3).

For more information on HDP user flash area errors, refer to Section 5.7: FLASH error management.
5.6 FLASH low-power modes

5.6.1 Introduction

Table 38 summarizes the behavior of the embedded flash memory in microcontroller low-power modes. The embedded flash memory belongs to the Core domain.

Table 38. Effect of low-power modes on the embedded flash memory

<table>
<thead>
<tr>
<th>Power mode</th>
<th>Core domain voltage range</th>
<th>Allowed if FLASH busy</th>
<th>FLASH power mode (in Core domain)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>VOS low / high</td>
<td>Yes</td>
<td>Run</td>
</tr>
<tr>
<td>Stop</td>
<td>SVOS low / high</td>
<td>No</td>
<td>Clock gated or Stopped</td>
</tr>
<tr>
<td>Standby</td>
<td>Off</td>
<td>No</td>
<td>Off</td>
</tr>
</tbody>
</table>

When the system state changes, or within a given system state, the embedded flash memory might be subjected to a different voltage supply range (VOS) depending on the application. The procedure to switch the embedded flash memory into various power modes (run, clock gated, stopped, off) is described below.

Note: For more information on microcontroller power states, refer to the Power control section (PWR).

5.6.2 Managing the FLASH domain switching to Stop or Standby

As explained in Table 38, if the embedded flash memory informs the reset and clock controller (RCC) that it is busy (that is, BUSY, QW, WBNE is set), the microcontroller cannot switch the Core domain to Stop or Standby mode.

Note: CRC_BUSY is not taken into account.

There are two ways to release the embedded flash memory:

- Reset the WBNE busy flag in the FLASH_SR register by any of the following actions:
  - Complete the write buffer with missing data.
  - Force the write operation without filling-in the missing data, by activating the FW bit in the FLASH_CR register. This forces all missing data “high”.
  - Reset the PG bit in FLASH_CR register. This disables the write buffer and consequently leads to the loss of its content.

- Poll the QW busy bits in the FLASH_SR register until they are cleared. This indicates that all recorded write, erase and option change operations are complete.

The microcontroller can then switch the domain to Stop or Standby mode.
5.7 FLASH error management

5.7.1 Introduction

The embedded flash memory automatically reports when an error occurs during a read, program or erase operation. A wide range of errors are reported:

- Write protection error (WRPERRF)
- Programming sequence error (PGSERRF)
- Strobe error (STRBERRF)
- Inconsistency error (INCERRF)
- Error correction code error (SNECCERRF/DBECCERRF)
- Read secure error (RDSERRF)
- CRC read error (CRCRDERRF)
- Option byte change error (OPTERRF)
- Key valid error (KVEF)
- Key transfer error (KTEF)
- Option byte loading error (OBLERRF)

The application software can individually enable the interrupt for each error, as detailed in Section 5.8: FLASH interrupts.

Note: For some errors, the application software must clear the error flag before attempting a new operation.

Note: General purpose errors (e.g. INCERRF) are taken into account when queues are loaded. Temporal isolation and other protection errors (for example WRPERRF, RDSERRF) are taken into account at FSM level, in front of the flash bank interface controller.

5.7.2 Write protection error (WRPERRF)

When an illegal erase/program operation is attempted to the non-volatile memory, the embedded flash memory sets the write protection error flag WRPERRF in FLASH_ISR register.

An erase operation is rejected and flagged as illegal if it targets one of the following memory areas:

- a sector belonging to a valid HDP user flash area (even partially), except if HDPL=0 or 1 in SBS_HDPLSR register.
- a sector write-locked with WRPS
- the read-only/OTP area

An program operation is ignored and flagged as illegal if it targets one of the following memory areas:

- a protected user flash sector belonging to a valid HDP user flash area (see Section 5.5.5)
- a user sector write-locked with WRPS
- an OTP block, locked with OTPL
- a read-only section
- the system flash memory
- a reserved area
When the WRPERRF flag is raised, the operation is rejected and nothing is changed. If a write burst operation was ongoing, WRPERRF is raised each time a flash word write operation is processed by the embedded flash memory.

**Note:** The WRPERRF flag does not block any new erase/program operation.

Not clearing the WRPERRF flag does not generate a programming sequence error (see below).

WRPERRF flag is cleared by setting the corresponding bit in FLASH_ICR register.

If WRPERRIE bit in FLASH_IER register is set, an interrupt is generated when the WRPERRF flag is raised (see Section 5.8: FLASH interrupts for details).

### 5.7.3 Programming sequence error (PGSERRF)

When the programming sequence is incorrect, the embedded flash memory sets the programming sequence error flag PGSERRF in FLASH_ISR register.

More specifically, the PGSERRF flag is set if one of the conditions below is met:

- A write operation is requested but the program enable bit (PG or PG_OTP) has not been set in the FLASH_CR register prior to the request. This program enable bit must stay at 1 until write is added to operation queue (QW rises).
- The inconsistency error (INCERRF) has not been cleared before requesting a new write operation.

When THE PGSERRF flag is raised, the current program operation is aborted and nothing is changed in flash. The corresponding write data buffer is also flushed. If a write burst operation was ongoing, PGSERRF is raised at the end of the burst.

**Note:** When the PGSERRF flag is raised, there is a risk that the last write operation performed by the application has been lost because of the above protection mechanism. Hence it is recommended to generate interrupts on PGSERRF and verify in the interrupt handler if the last write operation has been successful by reading back the value in the flash memory.

The PGSERRF flag also blocks any new program operation. This means that PGSERRF must be cleared before starting a new program operation.

The PGSERRF flag is cleared by setting the corresponding bit in the FLASH_ICR register.

If the PGSERRIE bit in FLASH_IER register is set, an interrupt is generated when the PGSERRF flag is raised. See Section 5.8: FLASH interrupts for details.

### 5.7.4 Strobe error (STRBERRF)

When the application software writes several times to the same byte write buffer, the embedded flash memory sets the strobe error flag STRBERRF in the FLASH_ISR register.

When the STRBERRF flag is raised, the current program operation is not aborted and new byte data replaces the old data. The application can ignore the error, proceed with the current write operation and request new write operations. If a write burst was ongoing, STRBERRF is raised at the end of the burst.

The STRBERRF flag is cleared by setting corresponding bit in the FLASH_ICR register.

If the STRBERRIE bit in FLASH_IER register is set, an interrupt is generated when the STRBERRF flag is raised. See Section 5.8: FLASH interrupts for details.
5.7.5 Inconsistency error (INCERRF)

When a programming inconsistency is detected, the embedded flash memory sets the inconsistency error flag INCERRF in the FLASH_ISR register.

More specifically, the INCERRF flag is set when one of the following conditions is met:

- A write operation is attempted before completion of the previous write operation, for example:
  - The application software starts a write operation to fill the 128-bit write buffer, but sends a new write burst request to a different flash memory address before the buffer is full.
  - One master starts a write operation, but before the buffer is full, another master starts a new write operation to the same address or to a different address.
- A wrap burst request issued by a master overlaps two or more 128-bit flash-word addresses, that is, wrap bursts must be done within 128-bit flash-word address boundaries.

Following the sequence below is recommended to avoid losing data when an inconsistency error occurs:

1. Execute a handler routine when the INCERRF flag is raised.
2. Stop all write requests to embedded flash memory.
3. Verify that the write operations that have been requested just before the INCERRF event have been successful by reading back the programmed values from the memory.
4. Clear the INCERRF bit in the FLASH_ICR register.
5. Restart the write operations where they have been interrupted.

Note: INCERRF flag must be cleared before starting a new write operation, otherwise a sequence error (PGSERRF) is raised.

Any write triggering inconsistency error is discarded by embedded flash memory.

5.7.6 Error correction code error (SNECCERRF/DBECCERRF)

When a single error correction is detected during a read the embedded flash memory sets the single error correction flag SNECCERRF in the FLASH_ISR register.

When two ECC errors are detected during a read, the embedded flash memory sets the double error detection flag DBECCERRF in the FLASH_ISR register.

When the SNECCERRF flag is raised, the corrected read data are returned. Hence the application can ignore the error and request new read operations.

If a read burst operation was ongoing, the SNECCERRF or DBECCERRF flag is raised each time a new data is sent back to the requester through the corresponding system bus.

When the SNECCERRF (or DBECCERRF respectively) flag is raised, the address of the flash word that generated the error is saved in the FLASH_ECCSFAR (or FLASH_ECCDFAR respectively) register. This register is automatically cleared when the associated flag that generated the error is cleared.

Note: In the case of successive single correction (or double detection respectively) errors, only the address corresponding to the first error is stored in the FLASH_ECCSFAR (or FLASH_ECCDFAR respectively) register.
When the DBECCERRF flag is raised reading the user flash memory, a bus error is generated. In the case of successive double error detections, a bus error is generated each time a new data is sent back to the requester through the AXI interface.

**Note:** It is not mandatory to clear the SNECCERRF or DBECCERRF flags before starting a new read operation.

The SNECCERRF and DBECCERRF flags are cleared by setting corresponding bit in the FLASH_ICR register.

If the SNECCERRIE (or respectively DBECCERRIE) bit in the FLASH_ISR register is set, an interrupt is generated when the SNECCERRF (or respectively DBECCERRF) flag is raised. See Section 5.8: FLASH interrupts for details.

### 5.7.7 Read secure error (RDSERRF)

When an illegal read or execute operation is attempted to a hide protected area, the embedded flash memory sets the read secure the error flag RDSERRF in the FLASH_ISR register. For more information on illegal access definition, refer to Section 5.5.4: Hide protected system flash area and Section 5.5.5: Hide protected user flash area.

When the RDSERRF flag is raised, the current read operation is aborted and the application can request new read operations. If a read burst was ongoing, RDSERRF is raised each time a data is sent back to the requester through the AXI interface.

**Note:** The bus error is raised only if the illegal access is due to an instruction fetch.

RDSERRF flag is cleared by setting corresponding bit in the FLASH_ICR register.

If the RDSERRIE bit in the FLASH_IER register is set, an interrupt is generated when the RDSERRF flag is raised (see Section 5.8: FLASH interrupts for details).

### 5.7.8 CRC read error (CRCRDERRF)

After a CRC computation, the embedded flash memory sets the CRC read error flag CRCERR in the FLASH_ISR register when one or more address belonging to a protected area was read by the CRC module. A protected area corresponds to a valid HDP area in user flash, for which any read is currently illegal (see Section 5.5.5 for details).

The CRCRDERRF flag is raised when the CRCEND bit is set (end of CRC calculation). In this case, it is likely that the CRC result is wrong since illegal read operations to protected areas return null values.

The CRCRDERRF flag is cleared by setting corresponding bit in the FLASH_ICR register.

If the CRCRDERRIE bit in the FLASH_IER register is set, an interrupt is generated when the CRCRDERRF flag is raised together with the CRCEND bit (see Section 5.8: FLASH interrupts for details).

### 5.7.9 Option byte change error (OPTERRF)

When the embedded flash memory finds an error during an option change operation, it aborts the operation and sets the option byte change the error flag OPTERRF in the FLASH_OPTISR register.

The OPTERRF flag is cleared by setting corresponding bit in the FLASH_OPTICR register.
If the OPTERRIE bit in the FLASH_OPTCR register is set, an interrupt is generated when the OPTERRF flag is raised (see Section 5.8: FLASH interrupts for details).

It is mandatory to clear the OPTERRF flag before starting a new option byte change, or an option byte key programming.

### 5.7.10 Key valid error (KVEF)

KVEF error flag is set in the FLASH_OPTISR register in two cases:

- Embedded flash did not find an option byte key that corresponds to the given OBKINDEX[4:0] and the requested HDPL (optionally modified by NEXTKL[1:0]). It can happen for example when requested key has not being provisioned.
- A double error detection was found when loading the requested option byte key. In this case, if this key is provisioned again the error should disappear.

It is mandatory to clear the KVEF flag before doing any operation on option byte keys because when KVEF is set, writes to the START bit in the FLASH_OBKCR are ignored.

The KVEF flag is cleared by setting corresponding bit in FLASH_OPTICR register.

If the KVEIE bit in the FLASH_OPTCR register is set, an interrupt is generated when the KVEF flag is raised (see Section 5.8: FLASH interrupts for details).

### 5.7.11 Key transfer error (KTEF)

The KTEF error flag is set in the FLASH_OPTISR register when embedded flash signals an error to the SAES peripheral. This happens when:

- the key size (128-bit or 256-bit) is not matching between embedded flash OBKSIZE[1:0] and KEYSIZE bit in SAES_CR register.
- an ECC dual error detection occurred while embedded flash loaded an option byte key for the SAES peripheral. In this case KVEF is also set.

It is mandatory to clear the KTEF flag before doing any operation on option byte keys. Because when KTEF is set write to the START bit in the FLASH_OBKCR is ignored.

The KTEF flag is cleared by setting corresponding bit in the FLASH_OPTICR register.

If the KTEIE bit in the FLASH_OPTCR register is set, an interrupt is generated when the KTEF flag is raised (see Section 5.8: FLASH interrupts for details).

While flash memory loads the option byte key to share with the SAES peripheral no other flash operation should be started until the key is transferred to the SAES. Otherwise the key transfer may fail (BUSY=KEYVALID=0 in the SAES_SR register) or the BUSY bit in the SAES_SR register stays at 1, until IPRST bit is used to clear it.

*Note:* When KTEF is cleared by the application the error signal to SAES is cleared.

### 5.7.12 Option byte loading error (OBLERRF)

When the embedded flash memory finds one of the following error during an option byte loading sequence (see Section 5.4.2) it sets the option byte change error flag OBLERRF in the FLASH_ISR register.

- ECC double error detection on one or more option byte (data or metadata)
- ECC double error detection on one or more option byte key (data or metadata)
In the case of an OBLERRF event the application should verify the correctness of the option byte information described in Section 5.4.

The OBLERRF flag is cleared by setting the corresponding bit in the FLASH_ICR register. If the OBLERRIE bit in the FLASH_IER register is set, an interrupt is generated when the OBLERRF flag is raised (see Section 5.8: FLASH interrupts for details).

### 5.7.13 Miscellaneous HardFault errors

The following events generate a bus error on the corresponding bus interface:

- On the AXI system bus:
  - illegal instruction fetch to HDP protected area because of the lifecycle (e.g. protected system flash when NVSTATE=OPEN).
- On AHB configuration or system bus:
  - wrong key input to FLASH_KEYR or FLASH_OPTKEYR
  - 8-bit accesses to system AHB interface

### 5.8 FLASH interrupts

The embedded flash memory can generate a maskable interrupt to signal the following events:

- Read and write errors (see Section 5.7: FLASH error management)
  - Single ECC error correction during read operation
  - Double ECC error detection during read operation
  - Write inconsistency error
  - Bad programming sequence
  - Strobe error during write operations
  - Option change error
  - Option byte keys error
- Security errors (see Section 5.7: FLASH error management)
  - Write protection error
  - Read security error
  - CRC computation on HDP area error
  - Option byte loading error
  - Key valid or key transfer errors
- Miscellaneous events (described below)
  - End of programming
  - CRC computation complete

These multiple sources are combined into a single interrupt signal, flash_it, which is the only interrupt signal from the embedded flash memory that drives the NVIC (nested vectored interrupt controller).

You can individually enable or disable embedded flash memory interrupt sources by changing the mask bits in the FLASH_IER register. Setting the appropriate mask bit enables the interrupt.
Note: Prior to writing, FLASH_IER register must be unlocked as explained in Section 5.5.1: FLASH configuration protection

Table 39 gives a summary of the available embedded flash memory interrupt features.

Note: Some flags need to be cleared before a new operation is triggered.

Table 39. Flash interrupt request

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear flag to resume operation</th>
<th>Bus error</th>
</tr>
</thead>
<tbody>
<tr>
<td>End-of-program event</td>
<td>EOPF</td>
<td>EOPIE</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>CRC complete event</td>
<td>CRCENDF</td>
<td>CRCENDIE</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Write protection error</td>
<td>WRPERRF</td>
<td>WRPERRIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Programming sequence error</td>
<td>PGSERRF</td>
<td>PGSERRIE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Strobe error</td>
<td>STRBERRF</td>
<td>STRBERRIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Inconsistency error</td>
<td>INCERRF</td>
<td>INCERRIE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>ECC single error correction</td>
<td>SNECCERRF</td>
<td>SNECCERRIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>ECC double error detection</td>
<td>DBECCERRF</td>
<td>DBECCERRIE</td>
<td>No</td>
<td>Yes(1)</td>
</tr>
<tr>
<td>Read secure error</td>
<td>RDSERRF</td>
<td>RDSERRIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>CRC error</td>
<td>CRCRDERRF</td>
<td>CRCRDERRIE</td>
<td>No</td>
<td>Yes(2)</td>
</tr>
<tr>
<td>Option byte loading error</td>
<td>OBLERRF</td>
<td>OBLERRIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Option byte change error</td>
<td>OPTERRF(2)</td>
<td>OPTERRIE</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Key valid error</td>
<td>KVEF(2)</td>
<td>KVEIE</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Key transfer error</td>
<td>KTEF(2)</td>
<td>KTEIE</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

1. AXI bus only
2. Set in FLASH_OPTISR register, cleared in FLASH_OPTICR register.

The status of the individual maskable interrupt sources described in Table 39 (except for option byte error) can be read from the FLASH_ISR register. They can be cleared by setting the corresponding bit in the FLASH_ICR register.

No unlocking mechanism is required to clear an interrupt.

**End-of-program event**

Setting the end-of-operation interrupt enable bit (EOPIE) in the FLASH_IER register enables the generation of an interrupt at the end of an erase operation, a program operation or an option byte change. The EOPF bit in the FLASH_ISR register is also set when one of these events occurs.

Setting the EOPF bit in FLASH_ICR register clears the EOPF flag in the FLASH_ISR register.
### CRC end of calculation event

Setting the CRC end-of-calculation interrupt enable bit (CRCENDIE) in the FLASH_IER register enables the generation of an interrupt at the end of a CRC operation. The CRCENDF bit in the FLASH_ISR register is also set when this event occurs.

Setting the CRCENDF bit in the FLASH_ICR register clears the CRCENDF flag in the FLASH_ISR register.

### 5.9 FLASH registers

#### 5.9.1 FLASH access control register (FLASH_ACR)

Address offset: 0x000  
Reset value: 0x0000 0013

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>WRHIGHFREQ</td>
<td>LATENCY[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 5:4 **WRHIGHFREQ[1:0]:** Flash signal delay  
These bits are used to control the delay between non-volatile memory signals during programming operations. Application software has to program them to the correct value depending on the embedded flash memory interface frequency. Refer to Table 29 for details.

*Note: Embedded flash does not verify that the configuration is correct.*

Bits 3:0 **LATENCY[3:0]:** Read latency  
These bits are used to control the number of wait states used during read operations on both non-volatile memory banks. The application software has to program them to the correct value depending on the embedded flash memory interface frequency and voltage conditions. Refer to Table 29 for details.

- 0000: zero wait state used to read a word from non-volatile memory  
- 0001: one wait state used to read a word from non-volatile memory  
- 0010: two wait states used to read a word from non-volatile memory  
- 0011: seven wait states used to read a word from non-volatile memory  
- 1111: fifteen wait states used to read from non-volatile memory  

*Note: Embedded flash does not verify that the configuration is correct.*
5.9.2 **FLASH control key register (FLASH_KEYR)**

Address offset: 0x004

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>CUKEY[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0 **CUKEY[31:0]**: Control unlock key

Following values must be written to FLASH_KEYR consecutively to unlock FLASH_CR register:

1st key = 0x4567 0123

2nd key = 0xCDEF 89AB

Reads to this register returns zero. If above sequence is wrong or performed twice, the FLASH_CR register is locked until the next system reset, and access to it generates a bus error.

5.9.3 **FLASH control register (FLASH_CR)**

Address offset: 0x010

Reset value: 0x0000 0001

When LOCK bit in this register is set writes to all other bits are ignored.

Write access to this register is controlled by FLASH_KEYR.

<table>
<thead>
<tr>
<th>Bits 31:25 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 24 <strong>ALL_BANKS</strong>: All banks select bit</td>
</tr>
<tr>
<td>When this bit is set the erase is done on all flash memory sectors.</td>
</tr>
<tr>
<td>ALL_BANKS is used only if a bank erase is required (BER=1). In all others operations, this control bit is ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:18 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17 <strong>CRC_EN</strong>: CRC enable</td>
</tr>
<tr>
<td>Setting this bit enables the CRC calculation. CRC_EN does not start CRC calculation but enables CRC configuration through FLASH_CRCRR register.</td>
</tr>
<tr>
<td>When CRC calculation is performed it can be disabled by clearing CRC_EN bit. Doing so sets CRCDATA to 0x0, clears CRC configuration and resets the content of FLASH_CRCDDAT register.</td>
</tr>
</tbody>
</table>
Bit 16 **PG_OTP**: Program Enable for OTP Area
Set this bit to enable write operations to OTP area.

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:6 **SSN[2:0]**: Sector erase selection number
These bits are used to select the target sector for an erase operation (they are unused otherwise).
- 000: Sector 0
- 001: Sector 1
- ...
- 111: Sector 7

Bit 5 **START**: Erase start control bit
This bit is used to start a sector erase or a bank erase operation. The embedded flash memory resets START when the corresponding operation has been acknowledged. The user application cannot access any embedded flash memory register until the operation is acknowledged.

Bit 4 **FW**: Force write
This bit forces a write operation even if the write buffer is not full. In this case all bits not written are set by hardware. The embedded flash memory resets FW when the corresponding operation has been acknowledged.

*Note: Using a force-write operation prevents the application from updating later the missing bits with something other than 1, because this would likely to lead to a permanent ECC error.*

Write forcing is effective only if the write buffer is not empty (in particular, FW does not start several write operations when the force-write operations are performed consecutively).

Bit 3 **BER**: Bank erase request
Setting this bit requests a bank erase operation (user flash memory only).

- 0: Bank erase is not requested
- 1: Bank erase is requested. Actual erase is started setting START bit in this register.

Write protection error is triggered when a bank erase is required and some sectors are protected.

BER has a higher priority than SER: if both are set, the embedded flash memory executes a bank erase.
Bit 2  **SER**: Sector erase request
   Setting this bit requests a sector erase.
   0: Sector erase not requested
   1: Sector erase requested
   Write protection error is triggered when a sector erase is required on at least one protected sector.
   BER has a higher priority than SER: if both bits are set, the embedded flash memory executes a bank erase.

Bit 1  **PG**: Internal buffer control bit
   Setting this bit enables internal buffer for write operations. This allows preparing program operations even if a sector or bank erase is ongoing. When PG is cleared, the internal buffer is disabled for write operations, and all the data stored in the buffer but not sent to the operation queue are lost.
   0: Internal buffer disabled for write operations
   1: Internal buffer enabled for write operations

Bit 0  **LOCK**: Configuration lock bit
   When this bit is set write to all other bits in this register, and to FLASH_IER register, are ignored.
   0: FLASH_CR and FLASH_IER registers are unlocked
   1: Writes to FLASH_IER, and to other bits than LOCK in FLASH_CR, are ignored
   Clearing this bit requires the correct write sequence to FLASH_KEYR register (see this register for details). If a wrong sequence is executed, or if the unlock sequence is performed twice, this bit remains locked until the next system reset.
   During the write access to set LOCK bit from 0 to 1, it is possible to change the other bits of this register.
### 5.9.4 FLASH status register (FLASH_SR)

Address offset: 0x014  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>IS_OPTCHANGE</td>
<td>IS_ERASE</td>
<td>IS_PROGRAM</td>
<td>CRC_BUSY</td>
<td>QW</td>
<td>WE</td>
<td>BUSY</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

**Bit 6 IS_OPTCHANGE:** Is an option change  
This bit is set together with BUSY when an option change operation is ongoing. It is cleared when BUSY is cleared.  
This flag can also raise with IS_PROGRAM or IS_ERASE, because a program or erase step is ongoing during option change.

**Bit 5 IS_ERASE:** Is an erase  
This bit is set together with BUSY when an erase operation is ongoing. It is cleared when BUSY is cleared.  
This flag can also raise with IS_OPTCHANGE, because an erase operation can happen during an option change.

**Bit 4 IS_PROGRAM:** Is a program  
This bit is set together with BUSY when a program operation is ongoing. It is cleared when BUSY is cleared.  
This flag can also raise with IS_OPTCHANGE, because a program operation can happen during an option change.

**Bit 3 CRC_BUSY:** CRC busy flag  
This bit is set when a CRC calculation is ongoing. This bit cannot be forced to 0. The user must wait until the CRC calculation has completed or disable CRC computation using CRC_EN bit in FLASH_CR register.  
0: No CRC calculation ongoing  
1: CRC calculation ongoing
Bit 2  **QW**: Wait queue flag
This bit is set when a write, erase or option byte change operation is pending in the command queue buffer. It is not possible to know what type of programming operation is present in the queue.
This flag is reset by hardware when all write, erase or option byte change operations have been executed and thus removed from the waiting queue(s). This bit cannot be forced to 0. It is reset after a deterministic time if no other operations are requested.
0: No write, erase or option byte change operations waiting in the operation queues
1: At least one write, erase or option byte change operation is waiting in the operation queue

Bit 1  **WBNE**: Write buffer not empty flag
This bit is set when the embedded flash memory is waiting for new data to complete the write buffer. In this state, the write buffer is not empty. WBNE is reset by hardware each time the write buffer is complete or the write buffer is emptied following one of the event below:
- the application software forces the write operation using FW bit in FLASH_CR
- the embedded flash memory detects an error that involves data loss
- the application software has disabled write operations
This bit cannot be forced to 0. To reset it, clear the write buffer by performing any of the above listed actions, or send the missing data.
0: Write buffer empty or full
1: Write buffer waiting data to complete

Bit 0  **BUSY**: Busy flag
This bit is set when an effective write, erase or option byte change operation is ongoing. It is possible to know what type of operation is being executed reading the flags IS_PROGRAM, IS_ERASE and IS_OPTCHANGE.
BUSY cannot be cleared by application. It is automatically reset by hardware every time a step in a write, erase or option byte change operation completes. It is not recommended to do software polling on BUSY to know when one operation completed because, depending of operation, more pulses are possible for one only operation. For software polling it is therefore better to use QW flag or to check the EOPF flag.
0: No programming, erase or option byte change operation are being executed
1: Programming, erase or option byte change operation are being executed. See flags IS_PROGRAM, IS_ERASE and IS_OPTCHANGE for details.
5.9.5  FLASH interrupt enable register (FLASH_IER)

Address offset: 0x020
Reset value: 0x0000 0000

Writes to this register are ignored if LOCK bit is set in FLASH_CR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>CRCRDERRIE</td>
<td>CRCENDIE</td>
<td>DBECCERRIE</td>
<td>SNECCERRIE</td>
<td>RDSERRIE</td>
<td>IncERRIE</td>
<td>OBLERRIE</td>
<td>STRBERRIE</td>
<td>PGSERRIE</td>
<td>WRPERRIE</td>
<td>EOPIE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 CRCRDERRIE: CRC read error interrupt enable bit
0: No interrupt is generated when CRCRDERRF bit is set in FLASH_ISR register
1: An interrupt is generated when CRCRDERRF bit is set in FLASH_ISR register

Bit 27 CRCENDIE: CRC end of calculation interrupt enable bit
0: No interrupt is generated when CRCEN bit is set in FLASH_ISR register
1: An interrupt is generated when CRCEN bit is set in FLASH_ISR register

Bit 26 DBECCERRIE: ECC double detection error interrupt enable bit
0: No interrupt is generated when DBECCERRF bit is set in FLASH_ISR register
1: An interrupt is generated when DBECCERRF bit is set in FLASH_ISR register

Bit 25 SNECCERRIE: ECC single correction error interrupt enable bit
0: No interrupt is generated when SNECCERRF bit is set in FLASH_ISR register
1: An interrupt is generated when SNECCERRF bit is set in FLASH_ISR register

Bit 24 RDSERRIE: Read security error interrupt enable bit
0: No interrupt is generated when RDSERRF bit is set in FLASH_ISR register
1: An interrupt is generated when RDSERRF bit is set in FLASH_ISR register

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 INCERRIE: Inconsistency error interrupt enable bit
0: No interrupt is generated when INCERRF bit is set in FLASH_ISR register
1: An interrupt is generated when INCERRF bit is set in FLASH_ISR register

Bit 20 OBLERRIE: Option byte loading error interrupt enable bit
0: No interrupt is generated when OBLERRF bit is set in FLASH_ISR register
1: An interrupt is generated when OBLERRF bit is set in FLASH_ISR register

Bit 19 STRBERRIE: Strobe error interrupt enable bit
0: No interrupt is generated when STRBERRF bit is set in FLASH_ISR register
1: An interrupt is generated when STRBERRF bit is set in FLASH_ISR register

Bit 18 PGSERRIE: Programming sequence error interrupt enable bit
0: No interrupt is generated when PGSERRF bit is set in FLASH_ISR register
1: An interrupt is generated when PGSERRF bit is set in FLASH_ISR register
Bit 17 **WRPERRIE**: Write protection error interrupt enable bit
0: No interrupt is generated when WRPERRF bit is set in FLASH_ISR register
1: An interrupt is generated when WRPERRF bit is set in FLASH_ISR register

Bit 16 **EOPIE**: End-of-program interrupt control bit
0: No interrupt is generated when OEPF bit is set in FLASH_ISR register
1: An interrupt is generated when OEPF bit is set in FLASH_ISR register

Bits 15:0 Reserved, must be kept at reset value.

### 5.9.6 FLASH interrupt status register (FLASH_ISR)

Address offset: 0x024

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res</td>
</tr>
<tr>
<td>30</td>
<td>Res</td>
</tr>
<tr>
<td>29</td>
<td>Res</td>
</tr>
<tr>
<td>28</td>
<td>CRCRDERRF</td>
</tr>
<tr>
<td>27</td>
<td>CRCENDF</td>
</tr>
<tr>
<td>26</td>
<td>DBECCERRF</td>
</tr>
<tr>
<td>25</td>
<td>SNECCERRF</td>
</tr>
<tr>
<td>24</td>
<td>RDSEFF</td>
</tr>
<tr>
<td>23</td>
<td>Res</td>
</tr>
<tr>
<td>22</td>
<td>Res</td>
</tr>
<tr>
<td>21</td>
<td>INCERRF</td>
</tr>
<tr>
<td>20</td>
<td>OBLERRF</td>
</tr>
<tr>
<td>19</td>
<td>STBERRF</td>
</tr>
<tr>
<td>18</td>
<td>PGSERRF</td>
</tr>
<tr>
<td>17</td>
<td>WRPERRF</td>
</tr>
<tr>
<td>16</td>
<td>EOPF</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **CRCRDERRF**: CRC read error flag
This bit is set when a word is found read protected during a CRC operation. An interrupt is generated if CRCRDIE is set. Setting CRCRDERRF bit in FLASH_ICR register clears this bit.
0: No protected area detected inside addresses read by CRC
1: At least one protected area has been detected inside address read by CRC. As a result CRC result is very likely incorrect.
This flag is valid only when CRCEND bit is set.

Bit 27 **CRCENDF**: CRC end flag
This bit is set when the CRC computation has completed. An interrupt is generated if CRCENDIE is set. It is not necessary to reset CRCEND before restarting CRC computation.
Setting CRCENDF bit in FLASH_ICR register clears this bit.
0: CRC computation not complete
1: CRC computation complete

Bit 26 **DBECCERRF**: ECC double error flag
This bit is set when an ECC double detection error occurs during a read operation. An interrupt is generated if DBECCERRIE is set. Setting DBECCERRF bit in FLASH_ICR register clears this bit.
0: No ECC double detection error occurred
1: ECC double detection error occurred
Embedded flash memory (FLASH)  

Bit 25  **SNECCERRF**: ECC single error flag  
This bit is set when an ECC single correction error occurs during a read operation. An interrupt is generated if SNECCERRIE is set. Setting SNECCERRF bit in FLASH_ICR register clears this bit.  
0: No ECC single correction error occurred  
1: ECC single correction error occurred  

Bit 24  **RDSERRF**: Read security error flag  
This bit is set when a read security error occurs (read access to hide protected area with incorrect hide protection level). An interrupt is generated if RDSERRIE is set. Setting RDSERRF bit in FLASH_ICR register clears this bit.  
0: No security error occurred  
1: A security error occurred  

Bits 23:22  Reserved, must be kept at reset value.  

Bit 21  **INCERRF**: Inconsistency error flag  
This bit is set when a inconsistency error occurs. An interrupt is generated if INCERRIE is set. Setting INCERRF bit in the FLASH_ICR register clears this bit.  
0: No inconsistency error occurred  
1: A inconsistency error occurred  

Bit 20  **OBLERRF**: Option byte loading error flag  
This bit is set when an error is found during the option byte loading sequence. An interrupt is generated if OBLERRIE is set. Setting OBLERRF bit in the FLASH_ICR register clears this bit.  
0: No error found during option byte loading sequence  
1: Some errors found during option byte loading sequence  

Bit 19  **STRBERRF**: Strobe error flag  
This bit is set when a strobe error occurs (when the master attempts to write several times the same byte in the write buffer). An interrupt is generated if the STRBERRIE bit is set. Setting STRBERRF bit in FLASH_ICR register clears this bit.  
0: No strobe error occurred  
1: A strobe error occurred  

Bit 18  **PGSERRF**: Programming sequence error flag  
This bit is set when a sequence error occurs. An interrupt is generated if the PGSERRIE bit is set. Setting PGSERRF bit in FLASH_ICR register clears this bit.  
0: No sequence error occurred  
1: A sequence error occurred  

Bit 17  **WRPERRF**: Write protection error flag  
This bit is set when a protection error occurs during a program operation. An interrupt is also generated if the WRPERRIE is set. Setting WRPERRF bit in FLASH_ICR register clears this bit.  
0: No write protection error occurred  
1: A write protection error occurred  

Bit 16  **EOPF**: End-of-program flag  
This bit is set when a programming operation completes. An interrupt is generated if the EOPIE is set. It is not necessary to reset EOPF before starting a new operation. Setting EOPF bit in FLASH_ICR register clears this bit.  
0: No programming operation completed  
1: A programming operation completed  

Bits 15:0  Reserved, must be kept at reset value.
5.9.7 **FLASH interrupt clear register (FLASH_ICR)**

Address offset: 0x028  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 28   | **CRCRDERRF**: CRC error flag clear  
Setting this bit clears CRCRDERRF flag in FLASH_ISR register. |
| 27   | **CRCENDF**: CRC end flag clear   
Setting this bit clears CRCENDF flag in FLASH_ISR register. |
| 26   | **DBECCERRF**: ECC double error flag clear  
Setting this bit clears DBECCERRF flag in FLASH_ISR register. If the SNECCERRF flag of FLASH_ISR register is also cleared, FLASH_ECCFAR register is reset to zero as well. |
| 25   | **SNECCERRF**: ECC single error flag clear  
Setting this bit clears SNECCERRF flag in FLASH_ISR register. If the DBECCERRF flag of FLASH_ISR register is also cleared, FLASH_ECCFAR register is reset to zero as well. |
| 24   | **RDSERRF**: Read security error flag clear  
Setting this bit clears RDSERRF flag in FLASH_ISR register. |
| 23:22 | Reserved, must be kept at reset value. |
| 21   | **INCERRF**: Inconsistency error flag clear  
Setting this bit clears INCERRF flag in FLASH_ISR register. |
| 20   | **OBLERRF**: Option byte loading error flag clear  
Setting this bit clears OBLERRF flag in FLASH_ISR register. |
| 19   | **STRBERRF**: Strobe error flag clear  
Setting this bit clears STRBERRF flag in FLASH_ISR register. |
| 18   | **PGSERRF**: Programming sequence error flag clear  
Setting this bit clears PGSERRF flag in FLASH_ISR register. |
| 17   | **WRPERRF**: Write protection error flag clear  
Setting this bit clears WRPERRF flag in FLASH_ISR register. |
| 16   | **EOPF**: End-of-program flag clear  
Setting this bit clears EOPF flag in FLASH_ISR register. |
| 15:0 | Reserved, must be kept at reset value. |
5.9.8 FLASH CRC control register (FLASH_CRCCR)

Address offset: 0x030
Reset value: 0x001C 0000

Writes to this register are ignored if CRC_EN bit is cleared in FLASH_CR register.

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Description</th>
<th>Access</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:25</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>ALL_SECT: All sectors selection</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When this bit is set all the sectors in user flash are added to list of sectors on which the CRC is to be calculated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is cleared when CRC computation starts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21:20</td>
<td>CRC_BURST[1:0]: CRC burst size</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CRC_BURST bits set the size of the bursts that are generated by the CRC calculation unit. A flash word is 128-bit.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00: every burst has a size of 4 flash words (64 Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>01: every burst has a size of 16 flash words (256 Bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10: every burst has a size of 64 flash words (1 Kbytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>11: every burst has a size of 256 flash words (4 Kbytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:18</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>CLEAN_CRC: CRC clear bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Setting CLEAN_CRC to 1 clears the current CRC result stored in the FLASH_CRCDATAR register.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>START_CRC: CRC start bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>START_CRC bit triggers a CRC calculation using the current configuration. No CRC calculation can launched when an option byte change operation is ongoing because all read accesses to embedded flash memory registers are put on hold until the option byte change operation has completed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit is cleared when CRC computation starts.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15:12</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>CLEAN_SECT: CRC sector list clear bit</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>When this bit is set the list of sectors on which the CRC is calculated is cleared.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 10 **ADD_SECT**: CRC sector select bit
When this bit is set the sector whose number is written in CRC_SECT is added to the list of sectors on which the CRC is calculated.

Bit 9 **CRC_BY_SECT**: CRC sector mode select bit
When this bit is set the CRC calculation is performed at sector level, on the sectors present in the list of sectors. To add a sector to this list, use ADD_SECT and CRC_SECT bits. To clean the list, use CLEAN_SECT bit.
When CRC_BY_SECT is cleared the CRC calculation is performed on all addresses defined between start and end addresses defined in FLASH_CRCSADDR and FLASH_CRCEADDR registers.

Bits 8:3 Reserved, must be kept at reset value.

Bits 2:0 **CRC_SECT[2:0]**: CRC sector number
CRC_SECT is used to select one user flash sectors to be added to the list of sectors on which the CRC is calculated. The CRC can be computed either between two addresses (using registers FLASH_CRCSADDR and FLASH_CRCEADDR) or on a list of sectors using this register. If this latter option is selected, it is possible to add a sector to the list of sectors by programming the sector number in CRC_SECT and then setting ADD_SECT bit.
The list of sectors can be erased either by setting CLEAN_SECT bit or by disabling the CRC computation.
000: sector 0 for CRC
001: sector 1 for CRC
... 111: sector 7 for CRC

5.9.9 **FLASH CRC start address register (FLASH_CRCSADDR)**
Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CRC_S</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TART</td>
<td>START</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADDR[10]</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 16:6 **CRC_START_ADDR[10:0]**: CRC start address
This register is used when CRC_BY_SECT is cleared. It must be programmed to the address of the first flash word to use for the CRC calculation, done burst by burst.
CRC computation starts at an address aligned to the burst size defined in CRC_BURST of FLASH_CRCCCR register. Hence least significant bits [5:0] of the address are set by hardware to 0 (minimum burst size = 64 bytes).
The address is relative to the flash bank.

Bits 5:0 Reserved, must be kept at reset value.
5.9.10 FLASH CRC end address register (FLASH_CRCEADDR)

Address offset: 0x038
Reset value: 0x0000 0000

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

Bits 16:6 CRC_END_ADDR[10:0]: CRC end address

This register is used when CRC_BY_SECT is cleared. It must be programmed to the address of the flash word starting the last burst of the CRC calculation. The burst size is defined in CRC_BURST of FLASH_CRCCR register.

The least significant bits [5:0] of the address are set by hardware to 0 (minimum burst size=64 bytes). The address is relative to the flash bank.

Bits 5:0 Reserved, must be kept at reset value.

5.9.11 FLASH CRC data register (FLASH_CRCDATAR)

Address offset: 0x03C
Reset value: 0x0000 0000

Bits 31:0 CRC_DATA[31:0]: CRC result

This bitfield contains the result of the last CRC calculation. The value is valid only when CRC calculation completed (CRCENDF is set in FLASH_ISR register).

CRC_DATA is cleared when CRC_EN is cleared in FLASH_CR register (CRC disabled), or when CLEAN_CRC bit is set in FLASH_CRCCR register.
5.9.12  FLASH ECC single error fail address (FLASH_ECCSFADDR)

Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>SEC_FADD[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0  SEC_FADD[31:0]: ECC single error correction fail address
- When a single ECC error correction occurs during a read operation, the SEC_FADD bitfield contains the system bus address that generated the error.
- This register is automatically cleared when SNECCERRF flag that generated the error is cleared.
- Note that only the first address that generated an ECC single error correction error is saved in this register.

5.9.13  FLASH ECC double error fail address (FLASH_ECCDFADDR)

Address offset: 0x044
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>DED_FADD[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:0  DED_FADD[31:0]: ECC double error detection fail address
- When a double ECC detection occurs during a read operation, the DED_FADD bitfield contains the system bus address that generated the error.
- This register is automatically cleared when the DBECCERRF flag that generated the error is cleared.
- Note that only the first address that generated an ECC double error detection error is saved in this register.
5.9.14  **FLASH options key register (FLASH_OPTKEYR)**

Address offset: 0x100  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>OCUKKEY[31:16]</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:0  **OCUKKEY[31:0]**: Options configuration unlock key  
Following values must be written to FLASH_OPTKEYR consecutively to unlock FLASH_OPTCR register:  
1st key = 0x0819 2A3B  
2nd key = 0x4C5D 6E7F  
Reads to this register returns zero. If above sequence is performed twice locks up the corresponding register/bit until the next system reset, and generates a bus error.

5.9.15  **FLASH options control register (FLASH_OPTCR)**

Address offset: 0x104  
Reset value: 0x0000 0001

When OPTLOCK bit in this register is set writes to all other bits are ignored.  
Write access to this register is controlled by FLASH_OPTKEYR register.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>OPTE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **Reserved, must be kept at reset value.**

Bit 30  **OPTERRIE**: Option byte change error interrupt enable bit  
This bit controls if an interrupt has to be generated when an error occurs during an option byte change.  
0: no interrupt is generated when an error occurs during an option byte change  
1: an interrupt is generated when and error occurs during an option byte change

Bit 29  **Reserved, must be kept at reset value.**

Bit 28  **KTEIE**: Key transfer error interrupt enable bit  
This bit controls if an interrupt has to be generated when KTEF is set in FLASH_OPTISR.  
0: no interrupt is generated when a key transfer error occurs  
1: an interrupt is generated when a key transfer error occurs
Bit 27  **KVEIE**: Key valid error interrupt enable bit
This bit controls if an interrupt has to be generated when KVEF is set in FLASH_OPTISR.
0: no interrupt is generated when a key valid error occurs
1: an interrupt is generated when a key valid error occurs

Bits 26:2  Reserved, must be kept at reset value.

Bit 1  **PG_OPT**: Program options
0: Update operations to user option bytes and option byte keys do not start
1: Write operation to user option bytes and option byte keys is enabled

Bit 0  **OPTLOCK**: Options lock
When this bit is set write to all other bits in this register, and write to OTP words, option bytes and option bytes keys control registers, are ignored.
0: OTP words, FLASH_OPTCR, FLASH_OBKCR and FLASH_xxSRP registers are unlocked
1: Writes to OTP words, FLASH_OBKCR, FLASH_xxSRP and to other bits than OPTLOCK in FLASH_OPTCR, are ignored
Clearing this bit requires the correct write sequence to FLASH_OPTKEYR register (see this register for details). If a wrong sequence is executed, or the unlock sequence is performed twice, this bit remains locked until next system reset.
During the write access to set LOCK bit from 0 to 1, it is possible to change the other bits of this register.

### 5.9.16  FLASH options interrupt status register (FLASH_OPTISR)
Address offset: 0x108
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bit 30  **OPTERRF**: Option byte change error flag
When OPTERRF is set, the option byte change operation did not successfully complete. An interrupt is generated when this flag is raised if the OPTERRIE bit of FLASH_OPTCR register is set. Setting OPTERRF of register FLASH_OPTCR clears this bit.
0: No option byte change errors occurred
1: One or more errors occurred during an option byte change operation.

Bit 29  Reserved, must be kept at reset value.
Bit 28  **KTEF**: Key transfer error flag
This bit is set when embedded flash signals an error to the SAES peripheral. It happens when the key size (128-bit or 256-bit) is not matching between embedded flash OBKSIZE[1:0] and KEYSIZE bit in SAES_CR register. It also happen when an ECC dual error detection occurred while embedded flash loaded an option byte key for the SAES peripheral.

When KTEF is set write to START bit in FLASH_OBKCR is ignored.
An interrupt is generated when this flag is raised if the KTEIE bit of FLASH_OPTCR register is set. Setting KTEF bit of register FLASH_OPTICR clears this bit.

Bit 27  **KVEF**: Key valid error flag
This bit is set when loading an unknown or corrupted option byte key. More specifically:

- Embedded flash did not find an option byte key that corresponds to the given OBKINDEX[4:0] and the requested HDPL (optionally modified by NEXTKL[1:0]). It can happen for example when requested key has not being provisioned.
- A double error detection was found when loading the requested option byte key. In this case, if this key is provisioned again the error should disappear.

When KVEF is set write to START bit in FLASH_OBKCR is ignored.
An interrupt is generated when this flag is raised if the KVEIE bit of FLASH_OPTCR register is set. Setting KVEF bit of register FLASH_OPTICR clears this bit.

Bits 26:0  Reserved, must be kept at reset value.

### 5.9.17  FLASH options interrupt clear register (FLASH_OPTICR)

Address offset: 0x10C
Reset value: 0xXXXX XXXX

### Table: FLASH_OPTICR

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OPTERRF</td>
<td>KTEF</td>
<td>KVEF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bit 30  **OPTERRF**: Option byte change error flag
Set this bit to clear OPTERRF flag in FLASH_OPTISR register.

Bit 29  Reserved, must be kept at reset value.

Bit 28  **KTEF**: key transfer error flag
Set this bit to clear KTEF flag in FLASH_OPTISR register.

Bit 27  **KVEF**: key valid error flag
Set this bit to clear KVEF flag in FLASH_OPTISR register.

Bits 26:0  Reserved, must be kept at reset value.
5.9.18  FLASH option byte key control register (FLASH_OBKCR)

Address offset: 0x110
Reset value: 0x0000 0C00

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  KEYSTART: Key option start
This bit is used to start the option byte key operation defined by the PROG bit.
The embedded flash memory resets START when the corresponding operation has been acknowledged.

Bit 14  KEYPROG: Key program
This bit must be set to write option byte keys (keys are read otherwise).
0: Read key. Result of the operation is stored in FLASH_OBKDRx registers, if applicable.
1: Program key if PG_OPT is set in FLASH_OPTCR register, and KDREF flag is cleared in
FLASH_OPTISR register. Correct key information must be stored in FLASH_OBKDRx
register before setting PROG and START bits.

Bits 13:12  Reserved, must be kept at reset value.

Bits 11:10  OBKSIZE[1:0]: Option byte key size
Application must use this bitfield to specify how many bits must be used for the new key.
Embedded flash ignores OBKSIZE during read of option keys because size is stored with the
key.
00: Key size is 32 bits
01: Key size is 64 bits
10: Key size is 128 bits
11: Key size is 256 bits

Bits 9:8  NEXTKL[1:0]: Next key level
00: OBKINDEX represents the index of the option byte key stored for the hide protection level
indicated in SBS_HDPLSR.
01: OBKINDEX represents the index of the option byte key stored for the hide protection level
indicated in SBS_HDPLSR plus one (e.g. if HDPL=1 in SBS_HDPLR the key of level 2 is
selected).
10 or 11: reserved

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  OBKINDEX[4:0]: Option byte key index
This bitfield represents the index of the option byte key in a given hide protection level.
Reading keys with index lower that 8, the value is not be available in OBKDRx registers. It is
instead sent directly to SAES peripheral. All others keys can be read using OBKDRx registers.
Up to 32 keys can be provisioned per hide protection level (0, 1 or 2), provided there is enough
space left in the flash to store them.
5.9.19  FLASH option bytes key data register x (FLASH_OBKDRx)

Address offset: 0x118 + 0x4 * x (x=0 to 7)
Reset value: 0x0000 0000
Writes are ignored if KDREF is set in FLASH_OPTISR register.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>OBKDATA[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:0  OBKDATA[31:0]: option byte key data, bits [31+x:0+x]
Data register used in conjunction with FLASH_OBKCR register.
Reading this register (read value once), or incrementing HDPL value in SBS peripheral
automatically clears OBKDATA to 0x0. Writing this register prevents reading OBKDATA until option
byte key programming sequence is completed.

5.9.20  FLASH non-volatile status register (FLASH_NVSR)

Address offset: 0x200
Reset value: 0x0000 XXXX
This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>NVSTATE[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>r r r r</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  NVSTATE[7:0]: Non-volatile state
0xB4: OPEN device
0x51: CLOSED device
others: invalid configuration.
5.9.21 FLASH security status register programming (FLASH_NVSRP)

Address offset: 0x204

Reset value: 0x0000 XXXX (same as FLASH_NVSR)

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR register.

Write are ignored if HDPL is different from 0 or 1 in SBS_HDPLSR register.

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>NVSTATE[7:0]: Non-volatile state programming</td>
</tr>
<tr>
<td></td>
<td>Write to change corresponding bits in FLASH_NVSR register:</td>
</tr>
<tr>
<td>0xB4:</td>
<td>OPEN</td>
</tr>
<tr>
<td>0x51:</td>
<td>CLOSE</td>
</tr>
<tr>
<td>Actual option byte change from close to open is triggered only after memory clear hardware process is confirmed. When NVSTATE=0xB4 (resp. 0x51) writing any other value than 0x51 (resp. 0xB4) triggers an option byte change error (OPTERRF).</td>
<td></td>
</tr>
</tbody>
</table>
5.9.22 FLASH RoT status register (FLASH_ROTSR)

Address offset: 0x208
Reset value: 0x0000 XXXX

This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IROT_SELECT[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 31:24 IROT_SELECT[7:0]:** iRoT selection
  - This option is ignored for STM32H7R device's (user firmware is always selected).
  - For STM32H7S devices there are two options:
    - 0xB4: ST iRoT is selected at boot
    - 0x6A (default): OEM iRoT is selected at boot.

- **Bits 23:16 Reserved, must be kept at reset value.**

- **Bits 15:8 DBG_AUTH[7:0]:** Debug authentication method
  - 0xB4: Locked device (no debug allowed)
  - 0x51: Authentication method using ECDSA signature (NIST P256)
  - 0x8A: Authentication method using password
  - Any other value: no authentication method selected (different to 0xB4, 0x51, 0x8A).

- **Bits 7:0 OEM_PROVD[7:0]:** OEM provisioned device
  - 0xB4: Device has been provisioned by the OEM
  - Any other value: device is not provisioned by the OEM.
5.9.23  FLASH RoT status register programming (FLASH_ROTSRP)

Address offset: 0x20C
Reset value: 0x0000 XXXX (same as FLASH_ROTSR)

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.

Writes are ignored if HDPL is different from 0 in SBS_HDPLSR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:24  IROT_SELECT[7:0]: iRoT selection
This option is ignored for STM32H7R devices.
Write to change corresponding bits in FLASH_ROTSR register.

Bits 23:16  Reserved, must be kept at reset value.

Bits 15:8  DBG_AUTH[7:0]: Debug authentication method programming
Write to change corresponding bits in FLASH_ROTSR register.

Bits 7:0  OEM_PROVD[7:0]: OEM provisioned device
Write to change corresponding bits in FLASH_ROTSR register.

5.9.24  FLASH OTP lock status register (FLASH_OTPLSR)

Address offset: 0x210
Reset value: 0x0000 XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| OTPL[15:0] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  OTPL[15:0]: OTP lock n (n=0 to 15)
Block n corresponds to OTP 16-bit word 32 x n to 32 x n + 31.
OTPL[n] = 1 indicates that all OTP 16-bit words in OTP Block n are locked and can no longer be programmed.
OTPL[n] = 0 indicates that all OTP 16-bit words in OTP Block n are not locked and can still be modified.
5.9.25 **FLASH OTP lock status register programming (FLASH_OTPLSRP)**

Address offset: 0x214  
Reset value: 0x0000 XXXX (same as for FLASH_OTPLSR)

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**OTPL[15:0]**

Bits 31:16 Reserved, must be kept at reset value.  
Bits 15:0 **OTPL[15:0]:** OTP lock n programming (n=0 to 15)  
Write to change corresponding option byte bit in FLASH_OTPLSR.  
OTPL bits can be only be set, not cleared.

5.9.26 **FLASH write protection status register (FLASH_WRPSR)**

Address offset: 0x218  
Reset value: 0x0000 XXXX

This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**WRPS[7:0]**

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **WRPS[7:0]:** Write protection for sector n (n=0 to 7)  
This bit reflects the write protection status of user flash sector n  
0: sector n is write protected  
1: sector n is not write protected
5.9.27  **FLASH write protection status register programming (FLASH_WRPSRP)**

Address offset: 0x21C

Reset value: 0x0000 XXXX

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.

Writes are ignored if HDPL is different from 0 or 1 in SBS_HDPLSR register.

(The reset value is the same as for FLASH_WRPSR)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>WRPS[7:0]: Write protection for sector n programming (n=0 to 7)</td>
<td>Write to change corresponding bit in FLASH_WRPSR</td>
</tr>
</tbody>
</table>

5.9.28  **FLASH hide protection status register (FLASH_HDPSR)**

Address offset: 0x230

Reset value: 0x0000 XXXX

This read-only register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>24:8</td>
<td>HDP_AREA_END[8:0]</td>
<td></td>
</tr>
<tr>
<td>7:0</td>
<td>HDP_AREA_START[8:0]</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25  Reserved, must be kept at reset value.
5.9.29 FLASH hide protection status register programming (FLASH_HDPSRP)

Address offset: 0x234
Reset value: 0x0XXX 0000

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.

Write are ignored if HDPL is different from 0 or 1 in SBS_HDPLSR register.

(The reset value is the same as for FLASH_HDPSR.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 HDP_AREA_END[8:0]: Hide protection user flash area end programming
Write to change corresponding option byte bits in FLASH_HDPSR.
If HDP_AREA_END=HDP_AREA_START all the sectors are protected.
If HDP_AREA_END<HDP_AREA_START no sectors are protected.

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 HDP_AREA_START[8:0]: Hide protection user flash area start programming
Write to change corresponding option byte bits in FLASH_HDPSR.
If HDP_AREA_END=HDP_AREA_START all the sectors are protected.
If HDP_AREA_END<HDP_AREA_START no sectors are protected.
5.9.30  **FLASH epoch status register (FLASH_EPOCHSR)**

Address offset: 0x250  
Reset value: 0x0000 XXXX  
This read-only register reflects the current values of corresponding option bits.

![Epoch Status Register Bit Diagram]

Bits 31:24  Reserved, must be kept at reset value.  
Bits 23:0  **EPOCH[23:0]**: Epoch  
This value is distributed by hardware to the SAES peripheral.

5.9.31  **FLASH RoT status register programming (FLASH_EPOCHSRP)**

Address offset: 0x254  
Reset value: 0x0000 XXXX  
Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.  
Writes are ignored if HDPL is different from 0 in SBS_HDPLSR register.  
(The reset value is the same as for FLASH_EPOCHSR.)

![Epoch Programming Register Bit Diagram]

Bits 31:24  Reserved, must be kept at reset value.  
Bits 23:0  **EPOCH[23:0]**: Epoch programming  
Write to change corresponding bits in FLASH_EPOCHSR register.
5.9.32 FLASH option byte word 1 status register (FLASH_OBW1SR)

Address offset: 0x260
Reset value: 0xXXXX XXXX (see Option byte user words organization)

This register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **VDDIO_HSLV**: I/O High-Speed at Low-Voltage
This bit indicates that the product operates below 2.5 V.
0: Product working in the full voltage range, I/O speed optimization at low-voltage disabled
1: Product operating below 2.5 V, I/O speed optimization at low-voltage feature allowed

Bit 28 **PERSO_OK**: Personalization OK
This bit is set on STMicroelectronics production line.

Bits 27:19 Reserved, must be kept at reset value.

Bit 18 **IWDG_FZ_SDBY**: IWDG standby mode freeze
When set the independent watchdog IWDG is frozen in system Standby mode.
0: Independent watchdog frozen in Standby mode
1: Independent watchdog keep running in Standby mode

Bit 17 **IWDG_FZ_STOP**: IWDG stop mode freeze
When set the independent watchdog IWDG is frozen in system Stop mode.
0: Independent watchdog frozen in Stop mode
1: Independent watchdog keep running in Stop mode

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **XSPI2_HSLV**: XSPIM_P2 High-Speed at Low-Voltage
0: I/O XSPIM_P2 High-Speed option disabled
1: I/O XSPIM_P2 High-Speed option enabled

Bit 8 **XSPI1_HSLV**: XSPIM_P1 High-Speed at Low-Voltage
0: I/O XSPIM_P1 High-Speed option disabled
1: I/O XSPIM_P1 High-Speed option enabled

Bit 7 **NRST_STBY**: Reset on standby mode
0: Independent WDG generates a reset if STANDBY mode is requested
1: Independent WDG does not generate a reset if STANDBY mode is requested
5.9.33 **FLASH option byte word 1 status register programming (FLASH_OBW1SRP)**

Address offset: 0x264

Reset value: 0xFFFF XXXX (same as for FLASH_OBW1SR)

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

- **Bit 29 VDDIO_HSLV**: I/O High-Speed at Low-Voltage programming
  Write to change corresponding bit in FLASH_OBW1SR register.

Bits 28:19 Reserved, must be kept at reset value.

- **Bit 18 IWDG_FZ_SDBY**: IWDG standby mode freeze programming
  Write to change corresponding bit in FLASH_OBW1SR register.

- **Bit 17 IWDG_FZ_STOP**: IWDG stop mode freeze
  Write to change corresponding bit in FLASH_OBW1SR register.
Embedded flash memory (FLASH)

5.9.34 FLASH option byte word 2 status register (FLASH_OBW2SR)

Address offset: 0x268
Reset value: 0xXXXX XXXX (see Option byte user words organization)

This register reflects the current values of corresponding option bits.

<table>
<thead>
<tr>
<th>Bits 31:20</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 19:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 17:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 15:14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 13:12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 11:10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 9:8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 7:6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 5:4</td>
<td>DTCM_AXI_SHARE[2:0]: DTCM SRAM configuration</td>
</tr>
<tr>
<td>Bits 3:2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 1:0</td>
<td>ITCM_AXI_SHARE[2:0]: ITCM SRAM configuration</td>
</tr>
</tbody>
</table>

### Bits Descriptions:

- **Bit 9** XSPI2_HSLV: XSPI_M_P2 High-Speed at Low-Voltage programming
  Write to change corresponding bit in FLASH_OBW1SR register.

- **Bit 8** XSPI1_HSLV: XSPI_M_P1 High-Speed at Low-Voltage
  Write to change corresponding bit in FLASH_OBW1SR register.

- **Bit 7** NRST_STBY: Reset on standby mode programming
  Write to change corresponding bit in FLASH_OBW1SR register.

- **Bit 6** NRST_STOP: Reset on stop mode programming
  Write to change corresponding bit in FLASH_OBW1SR register.

- **Bit 5** Reserved, must be kept at reset value.

- **Bit 4** IWDG_HW: Independent watchdog HW Control
  Write to change corresponding bit in FLASH_OBW1SR register.

- **Bits 3:2** BOR_LEV[1:0]: Brownout level
  Write to change corresponding bits in FLASH_OBW1SR register.

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

---

STMicroelectronics

272/3791 RM0477 Rev 6
### 5.9.35 FLASH option byte word 2 status register programming (FLASH_OBW2SRP)

Address offset: 0x26C

Reset value: 0xXXXX XXXX (same as for FLASH_OBW2SR)

Writes to this register are ignored if PG_OPT bit is cleared in FLASH_OPTCR register, or if BUSY and IS_OPTCHANGE bits are set in FLASH_SR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- **Bit 9** I2C_NI3C: I2C Not I3C
  
  Write to change corresponding bit in FLASH_OBW2SR register.

- **Bit 8** ECC_ON_SRAM: ECC on SRAM programming
  
  Write to change corresponding bit in FLASH_OBW2SR register.

- **Bit 7** Reserved, must be kept at reset value.

- **Bits 6:4** DTCM_AXI_SHARE[2:0]: DTCM AXI share programming
  
  Write to change corresponding bits in the FLASH_OBW2SR register.
  
  Bit 2 should be kept to 0:
  
  - DTCM_AXI_SHARE = "000" or "011": DTCM 64 Kbytes
  - DTCM_AXI_SHARE = "001": DTCM 128 Kbytes
  - DTCM_AXI_SHARE = "010": DTCM 192 Kbytes

- **Bit 3** Reserved, must be kept at reset value.

- **Bits 2:0** ITCM_AXI_SHARE[2:0]: ITCM AXI share programming
  
  Write to change corresponding bits in FLASH_OBW2SR register.
  
  Bit 2 should be kept to 0:
  
  - ITCM_AXI_SHARE = "000" or "011": ITCM 64 Kbytes
  - ITCM_AXI_SHARE = "001": ITCM 128 Kbytes
  - ITCM_AXI_SHARE = "010": ITCM 192 Kbytes
## 5.9.36 FLASH register map

### Table 40. FLASH register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>FLASH_ACR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>FLASH_KEYR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CUKEY[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CUKEY[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>FLASH_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>FLASH_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>FLASH_IER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>FLASH_ISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>FLASH_ICR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>FLASH_CRCCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>FLASH_CRCSADDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Reset Values

- **FLASH_ACR**: 0x000
- **FLASH_KEYR**: 0x004
- **FLASH_CR**: 0x010
- **FLASH_SR**: 0x014
- **FLASH_IER**: 0x020
- **FLASH_ISR**: 0x024
- **FLASH_ICR**: 0x028
- **FLASH_CRCCR**: 0x030
- **FLASH_CRCSADDR**: 0x034
RM0477

Embedded flash memory (FLASH)

0

0

0

0

0

0

0

FLASH_ECCDFADD
R

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

Res.

CRC_END_ADDR[10]

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

Res.

Res.

Res.

0

0

Res.

Res.

0

0

Res.

Res.

0

0

0

0

0

0

0

0

0

0

0

0

0

0

FLASH_OBKDR3
Reset value

0

0

0

0

FLASH_OBKDR4
Reset value

0

0

0

0

0

0

0

0

0

0

0

0

Res.

Res.

Res.

Res.

0

0

0

0

0

0

0

0

0

0

PG_OPT

OPTLOCK

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

1

1

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

0

NVSTATE[7:0]
1

Res.

0

OBKDATA[15:0]

0

Res.

0

OBKDATA[15:0]

0

Res.

0

OBKDATA[15:0]

0

Res.

0

OBKDATA[15:0]

0

Res.

0

OBKDATA[15:0]

0

Res.

0

OBKDATA[15:0]

0

Res.

0

OBKDATA[15:0]

0

Res.

OBKINDEX[4:0]

NEXTKL[1:0]

OBKSIZE[1:0]
0

0

Res.

Res.

Res.
KEYPROG

Res.

Res.
KEYSTART

Res.

Res.
Res.

Res.

Res.
Res.

Res.

Res.
Res.

Res.

Res.
Res.

Res.

Res.
Res.

Res.

Res.
Res.

0

OBKDATA[31:16]

Res.

FLASH_NVSRP

0

0

Res.

FLASH_NVSR

0

OBKDATA[31:16]

FLASH_OBKDR7
Reset value

0

OBKDATA[31:16]

FLASH_OBKDR6
Reset value

0

OBKDATA[31:16]

FLASH_OBKDR5
Reset value

0

OBKDATA[31:16]
0

Res.

0

0

OBKDATA[15:0]

OBKDATA[31:16]
0

Res.

0

Res.

0

FLASH_OBKDR2
Reset value

1

OBKDATA[31:16]
0

1

Res.

0

FLASH_OBKDR1
Reset value

1

OBKDATA[31:16]
0

Res.

Res.
Res.

Res.

FLASH_OBKDR0
Reset value

Res.

Res.
Res.

0

Res.

Res.
Res.

0

0

X X

Res.

Res.

FLASH_OBKCR

X

Res.

FLASH_OPTICR

OCUKEY[15:0]

Res.

X X
KVEF

X
Res.

Reset value

KTEF

FLASH_OPTISR

OCUKEY[31:16]

Res.

Reset value

Res.

FLASH_OPTCR

Reset value
0x204

0

0

Res.

0x200

0

0

Res.

0x134

0

Reset value

Res.

0x130

0

DED_FADD[15:0]

Res.

0x12C

0

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0

0

SEC_FADD[15:0]

Res.

0

0

0

Res.

0

Res.

0

0

Res.

0

KVEIE

0

0

Res.

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0

0

Res.

0x128

Res.

0

Res.

0x124

Res.

0

Res.

0x120

Res.

0

Res.

0x11C

Res.

0

Reset value
0x118

Res.

0

DED_FADD[31:16]

Reset value

0x110

0

KTEIE

FLASH_OPTKEYR

0

SEC_FADD[31:16]

Res.

0x10C

0

Res.

0x108

0

OPTERRIE

0x104

0

OPTERRF

0x100

0

Res.

Reset value
0x044

0

FLASH_ECCSFADD
R

CRC_DATA[15:0]

Res.

Reset value

0

CRC_DATA[31:16]

Res.

0x040

0

FLASH_CRCDATAR

Res.

0x03C

Res.

Res.

Reset value

CRC_END_ADDR[10]

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

FLASH_CRCEADDR

Res.

0x038

Res.

Offset Register name

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

Table 40. FLASH register map and reset values (continued)

1

1

1

1

1

NVSTATE[7:0]

X X X X X X X X X X X X X X X X

RM0477 Rev 6

275/3791
277


### Table 40. FLASH register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x208</td>
<td>FLASH_ROTSR</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>IROT_SELECT[7:0]</td>
<td>26</td>
<td>Reserved</td>
<td>25</td>
<td>Reserved</td>
<td>24</td>
<td>Reserved</td>
<td>23</td>
<td>Reserved</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>1 1 1 0 1 0 1 0</td>
<td>21</td>
<td>Reserved</td>
<td>20</td>
<td>Reserved</td>
<td>19</td>
<td>Reserved</td>
<td>18</td>
<td>Reserved</td>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x20C</td>
<td>FLASH_ROTSRP</td>
<td>16</td>
<td>Reserved</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td>13</td>
<td>Reserved</td>
<td>12</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>IROT_SELECT[7:0]</td>
<td>11</td>
<td>Reserved</td>
<td>10</td>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>1 1 1 0 1 0 1 0</td>
<td>6</td>
<td>Reserved</td>
<td>5</td>
<td>Reserved</td>
<td>4</td>
<td>Reserved</td>
<td>3</td>
<td>Reserved</td>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x210</td>
<td>FLASH_OTPLSR</td>
<td>1</td>
<td>Reserved</td>
<td>0</td>
<td>Reserved</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>OTPL[15:0]</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
<td>26</td>
<td>Reserved</td>
<td>25</td>
<td>Reserved</td>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>X X X X X X X X X X X X X X X X</td>
<td>23</td>
<td>Reserved</td>
<td>22</td>
<td>Reserved</td>
<td>21</td>
<td>Reserved</td>
<td>20</td>
<td>Reserved</td>
<td>19</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x214</td>
<td>FLASH_OTPLSRP</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td>13</td>
<td>Reserved</td>
<td>12</td>
<td>Reserved</td>
<td>11</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>OTPL[15:0]</td>
<td>10</td>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>7</td>
<td>Reserved</td>
<td>6</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>X X X X X X X X X X X X X X X X</td>
<td>5</td>
<td>Reserved</td>
<td>4</td>
<td>Reserved</td>
<td>3</td>
<td>Reserved</td>
<td>2</td>
<td>Reserved</td>
<td>1</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x218</td>
<td>FLASH_WRPASR</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>WRP[7:0]</td>
<td>26</td>
<td>Reserved</td>
<td>25</td>
<td>Reserved</td>
<td>24</td>
<td>Reserved</td>
<td>23</td>
<td>Reserved</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>X X X X X X X X X X X X</td>
<td>21</td>
<td>Reserved</td>
<td>20</td>
<td>Reserved</td>
<td>19</td>
<td>Reserved</td>
<td>18</td>
<td>Reserved</td>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x21C</td>
<td>Reserved</td>
<td>16</td>
<td>Reserved</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td>13</td>
<td>Reserved</td>
<td>12</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x22C</td>
<td>Reserved</td>
<td>11</td>
<td>Reserved</td>
<td>10</td>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x230</td>
<td>Reserved</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x248</td>
<td>Reserved</td>
<td>26</td>
<td>Reserved</td>
<td>25</td>
<td>Reserved</td>
<td>24</td>
<td>Reserved</td>
<td>23</td>
<td>Reserved</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x250</td>
<td>FLASH_EPOCHSR</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X X X X X X X X X X X X X X X X X X X X X X</td>
<td>21</td>
<td>Reserved</td>
<td>20</td>
<td>Reserved</td>
<td>19</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x254</td>
<td>FLASH_EPOCHSRP</td>
<td>16</td>
<td>Reserved</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td>13</td>
<td>Reserved</td>
<td>12</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>EPOCH[23:16]</td>
<td>11</td>
<td>Reserved</td>
<td>10</td>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 X X X X X X X X X X X X X X X X X X X X X X</td>
<td>6</td>
<td>Reserved</td>
<td>5</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x260</td>
<td>FLASH_OBW1SR</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>VDDIO_HSLV</td>
<td>26</td>
<td>Reserved</td>
<td>25</td>
<td>Reserved</td>
<td>24</td>
<td>Reserved</td>
<td>23</td>
<td>Reserved</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>PERIO_OK</td>
<td>21</td>
<td>Reserved</td>
<td>20</td>
<td>Reserved</td>
<td>19</td>
<td>Reserved</td>
<td>18</td>
<td>Reserved</td>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>X X X X X X X X X X X X X X X X X X X X X X</td>
<td>16</td>
<td>Reserved</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x264</td>
<td>FLASH_OBW1SRP</td>
<td>16</td>
<td>Reserved</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td>13</td>
<td>Reserved</td>
<td>12</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>VDDIO_HSLV</td>
<td>11</td>
<td>Reserved</td>
<td>10</td>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>XSPI2_HSLV</td>
<td>6</td>
<td>Reserved</td>
<td>5</td>
<td>Reserved</td>
<td>4</td>
<td>Reserved</td>
<td>3</td>
<td>Reserved</td>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>X X X X X X X X X X X X X X X X X X X X X X</td>
<td>1</td>
<td>Reserved</td>
<td>0</td>
<td>Reserved</td>
<td>32</td>
<td>Reserved</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x268</td>
<td>FLASH_OBW2SR</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
<td>29</td>
<td>Reserved</td>
<td>28</td>
<td>Reserved</td>
<td>27</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>I2C_NIC</td>
<td>26</td>
<td>Reserved</td>
<td>25</td>
<td>Reserved</td>
<td>24</td>
<td>Reserved</td>
<td>23</td>
<td>Reserved</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>ECC_ON_SRAM</td>
<td>21</td>
<td>Reserved</td>
<td>20</td>
<td>Reserved</td>
<td>19</td>
<td>Reserved</td>
<td>18</td>
<td>Reserved</td>
<td>17</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value</td>
<td>X X X X X X X X X X X X X X X X X X X X X X</td>
<td>16</td>
<td>Reserved</td>
<td>15</td>
<td>Reserved</td>
<td>14</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x270</td>
<td>Reserved</td>
<td>11</td>
<td>Reserved</td>
<td>10</td>
<td>Reserved</td>
<td>9</td>
<td>Reserved</td>
<td>8</td>
<td>Reserved</td>
<td>7</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x274</td>
<td>Reserved</td>
<td>6</td>
<td>Reserved</td>
<td>5</td>
<td>Reserved</td>
<td>4</td>
<td>Reserved</td>
<td>3</td>
<td>Reserved</td>
<td>2</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x278</td>
<td>Reserved</td>
<td>1</td>
<td>Reserved</td>
<td>0</td>
<td>Reserved</td>
<td>32</td>
<td>Reserved</td>
<td>31</td>
<td>Reserved</td>
<td>30</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Reset values are shown in hexadecimal format.
Table 40. FLASH register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x26C</td>
<td>FLASH_OBW2SRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

Reset value

X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
6 Power control (PWR)

6.1 Introduction

The power control section (PWR) provides an overview of the supply architecture for the different power domains and of the supply configuration controller.

It also describes the features of the power supply supervisors and explains how the $V_{\text{CORE}}$ supply domain is configured depending on the operating modes, the selected performance (clock frequency) and the voltage scaling.

6.2 PWR main features

- Power supplies and supply domains
  - Core domains ($V_{\text{CORE}} = V_{\text{CAP}}$)
  - $V_{\text{DD}}$ domain
  - Backup domain ($V_{\text{SW}}$, $V_{\text{BKP}}$)
  - Analog domain ($V_{\text{DDA}}$)

- System supply voltage regulation
  - Switched-mode power supply power-efficient voltage down-converter (SMPS step-down converter)
  - Voltage regulator (LDO)

- Peripheral supply regulation
  - USB regulator

- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - PVD monitor
  - AVD monitor
  - $V_{\text{BAT}}$ thresholds
  - Temperature thresholds

- Power management
  - $V_{\text{BAT}}$ battery charging
  - Operating modes
  - Voltage scaling control
  - Low-power modes
6.3 PWR block diagram

Figure 19. Power control block diagram
6.3.1 PWR pins and internal signals

Table 41 lists the PWR inputs and output signals connected to package pins or balls, while Table 42 shows the internal PWR signals.

Table 41. PWR input/output signals connected to package pins or balls

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>Supply input</td>
<td>Main I/O and ( V_{DD} ) domain supply input</td>
</tr>
<tr>
<td>VDDA</td>
<td>Supply input</td>
<td>External analog power supply for analog peripherals</td>
</tr>
<tr>
<td>VSSA</td>
<td>Supply Analog peripherals ground</td>
<td></td>
</tr>
<tr>
<td>VREF+,VREF-</td>
<td>Supply input/output</td>
<td>External reference voltage for ADCs</td>
</tr>
<tr>
<td>VBAT</td>
<td>Supply input/output</td>
<td>Backup battery supply input</td>
</tr>
<tr>
<td>VDDSMPS</td>
<td>Supply input</td>
<td>Step-down converter supply input</td>
</tr>
<tr>
<td>VLXSMPS</td>
<td>Supply output</td>
<td>Step-down converter supply output</td>
</tr>
<tr>
<td>VFBSMPS</td>
<td>Supply input</td>
<td>Step-down converter feedback voltage sense</td>
</tr>
<tr>
<td>VSSSMPSE</td>
<td>Supply input</td>
<td>Step-down converter ground</td>
</tr>
<tr>
<td>VDDLDOS</td>
<td>Supply input</td>
<td>Voltage regulator supply input</td>
</tr>
<tr>
<td>VCAP</td>
<td>Supply Input/Outputs</td>
<td>Digital core domain supply</td>
</tr>
<tr>
<td>VDD50USB</td>
<td>Supply input</td>
<td>USB regulator supply input</td>
</tr>
<tr>
<td>VDD33USB</td>
<td>Supply Input/Outputs</td>
<td>USB regulator supply output or external USB supply input</td>
</tr>
<tr>
<td>DVDD</td>
<td>Supply Digital supply for the USB HS PHY</td>
<td></td>
</tr>
<tr>
<td>VSS</td>
<td>Supply Main ground</td>
<td></td>
</tr>
<tr>
<td>AHB</td>
<td>Input/output</td>
<td>AHB register interface</td>
</tr>
<tr>
<td>WKUPx</td>
<td>Input Wakeup pins</td>
<td></td>
</tr>
<tr>
<td>SLEEP</td>
<td>Output MCU in Sleep mode</td>
<td></td>
</tr>
<tr>
<td>STOP</td>
<td>Output MCU in Stop modes</td>
<td></td>
</tr>
</tbody>
</table>

Table 42. PWR internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AHB</td>
<td>Input/output</td>
<td>AHB register interface</td>
</tr>
<tr>
<td>pwr_pvd_wkup</td>
<td>Output</td>
<td>Programmable voltage detector output</td>
</tr>
<tr>
<td>pwr_avd_wkup</td>
<td>Output</td>
<td>Analog voltage detector output</td>
</tr>
<tr>
<td>pwr_por_rst</td>
<td>Output</td>
<td>Power-on reset</td>
</tr>
<tr>
<td>pwr_bor_rst</td>
<td>Output</td>
<td>Brownout reset</td>
</tr>
<tr>
<td>exti_wkup</td>
<td>Input</td>
<td>Wakeup request</td>
</tr>
<tr>
<td>pwr_wkup</td>
<td>Output</td>
<td>Bus matrix clock wakeup request</td>
</tr>
</tbody>
</table>
Each of the four wakeup events, WKUPx, can be generated from four pins or internal events.

<table>
<thead>
<tr>
<th>Port</th>
<th>Wakeup event</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA0</td>
<td>WKUP1</td>
</tr>
<tr>
<td>PA2</td>
<td>WKUP2</td>
</tr>
<tr>
<td>PC13</td>
<td>WKUP3</td>
</tr>
<tr>
<td>PC1</td>
<td>WKUP4</td>
</tr>
</tbody>
</table>
6.4 Power supplies

The device requires $V_{DD}$ and $V_{DDSMPS}$ power supplies as well as independent supplies for $V_{DDLDO}$, $V_{DDA}$, $V_{DDUSB}$, and $V_{CAP}$. It also provides regulated supplies for specific functions (step-down converter, voltage regulator, USB regulator).

- $V_{DD}$: external power supply for I/Os and system analog blocks such as reset, power management and oscillators
- $V_{DDxspi1}, V_{DDxspi2}$: external power supply for dedicated I/Os (Octo SPI and Hexa Deca SPI memories)
- $V_{BAT}$: optional external power supply for backup domain when $V_{DD}$ is not present ($V_{BAT}$ mode)
  This power supply must be connected to VDD when no battery is used.
- $V_{DDSMPS}$: external power supply for the SMPS step-down converter
  VDDSMS must be connected to VDD or tied to VSS when the SMPS is not used.
- $V_{LXSMPS}$: step-down converter supply output
- $V_{FBSMPS}$: step-down converter sense feedback
- $V_{SSSMPS}$: separate step-down converter ground
- $V_{DDLDO}$: external power supply for voltage regulator
- $V_{CAP}$: digital core domain supply
  This power supply is independent from all the other power supplies:
  – When the voltage regulator is enabled, $V_{CORE}$ is delivered by the internal voltage regulator.
  – When the voltage regulator is disabled, $V_{CORE}$ is delivered by an external power supply through VCAP pin, or by the SMPS step-down converter.
- $V_{DDA}$: external analog power supply for ADCs and voltage reference buffers
  This power supply is independent from all the other power supplies.
- $V_{REF+}$: external reference voltage for ADC.
  – When the voltage reference buffer is enabled, $V_{REF+}$ and $V_{REF-}$ are delivered by the internal voltage reference buffer.
  – When the voltage reference buffer is disabled, $V_{REF+}$ is delivered by an independent external reference supply.
- $V_{SSA}$: separate analog and reference voltage ground.
- $V_{DD50USB}$: external power supply for USB regulator.
- $V_{DD33USB}$: USB regulator supply output for USB interface.
  – When the USB regulator is enabled, $V_{DD33USB}$ is delivered by the internal USB regulator.
  – When the USB regulator is disabled, $V_{DD33USB}$ is delivered by an independent external supply input.
- $V_{SS}$: common ground for all supplies except for step-down converter and analog peripherals.

Note: Depending on the operating power supply range, some peripherals might be used with limited features and performance. For more details, refer to section “General operating conditions” of the device datasheets.
Figure 20. Power supply overview(a)

a. Refer to application note AN5935 "Getting started hardware with STM32H7Rx/7Sx MCUs" for the possible power scheme and connected capacitors.
By configuring the SMPS step-down converter and LDO voltage regulator, the supply configurations shown in Figure 21 are supported for the V\textsubscript{CORE} core domain and an external supply. For the different configuration available per package refer to the product datasheet.

Note: The SMPS step-down converter is not available on all packages, and the Bypass mode is available only when the SMPS is available.
The different supply configurations are controlled through the LDOEN, SDEN, SMPSEXTHP, SDHILEVEL and BYPASS bits in the PWR control register 2 (PWR_CSR2) register according to Table 44.

In the following table HP refers to High power, LP refers to Low Power and off refers to no supply.

### Table 44. Supply configuration control

<table>
<thead>
<tr>
<th>ID</th>
<th>Supply configuration</th>
<th>SDHILEVEL</th>
<th>SMPSEXTHP</th>
<th>SDEN</th>
<th>LDOEN</th>
<th>BYPASS</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0  | Startup configuration | 0         | 0         | 1    | 1     | 0      | - \( V_{\text{CORE}} \) power domains are supplied from the LDO according to VOS.  
- SMPS step-down converter enabled at 1.35 V, may be used to supply the LDO. |
| 1  | LDO supply           | x         | x         | 0    | 1     | 0      | - \( V_{\text{CORE}} \) power domains are supplied from the LDO according to VOS.  
- LDO power mode (HP, LP, Off) follows the system low-power modes.  
- SMPS step-down converter disabled. |
| 2  | Direct SMPS step-down converter supply | X         | 0         | 1    | 0     | 0      | - \( V_{\text{CORE}} \) power domains are supplied from SMPS step-down converter according to VOS.  
- LDO OFF.  
- SMPS step-down converter power mode (HP, LP, Off) follows system low-power modes.  
- SDHILEVEL is not taken into account, and is considered to be 0. |
| 3  | SMPS step-down converter supplies External. LDO supplied by external VDD. | 1         | 1         | 1    | 1     | 0      | - \( V_{\text{CORE}} \) power domains are supplied from voltage regulator according to VOS  
- LDO power mode (HP, LP, Off) follows system low-power modes.  
- SMPS step-down converter enabled according to SDHILEVEL used to supply external circuits.  
- SMPS step-down converter forced ON in HP mode. |
| 4  | SMPS step-down converter supplies external and LDO Bypass | 1         | 1         | 1    | 0     | 1      | - \( V_{\text{CORE}} \) supplied from external source  
- SMPS step-down converter enabled according to SDHILEVEL used to supply external circuits and may supply the external source for \( V_{\text{CORE}} \).  
- SMPS step-down converter forced ON in HP mode. |
| 5  | SMPS step-down converter disabled and LDO Bypass | x         | x         | 0    | 0     | 1      | - \( V_{\text{CORE}} \) supplied from external source  
- SMPS step-down converter disabled and LDO bypassed, voltage monitoring still active. |
6.4.1 System supply startup

The system startup sequence from power-on in different supply configurations is the following (see *Figure 22* and *Figure 23* for LDO supply and Direct SMPS supply, respectively):

1. When the system is powered on, the POR monitors \( V_{DD} \) supply. Once \( V_{DD} \) is above the POR threshold level, the SMPS step-down converter and voltage regulator are enabled in the default supply configuration:
   - The SMPS step-down converter output level is set at 1.35 V.
   - The voltage regulator LDO output level is set at 1.11 V.
   - Depending on the package and configuration the SMPS provides the voltage to internal power domain and/or LDO or external supply.
2. The system is kept in reset mode as long as \( V_{CORE} \) is not stable.
3. Once \( V_{CORE} \) is stable, the system is taken out of reset and the HSI oscillator is enabled.
4. Once the oscillator is stable, the system is initialized: Flash memory is ready, option bytes are loaded and the CPU starts in limited run mode (Run*).
5. The software must then initialize the system including supply configuration programming in *PWR control register 2 (PWR_CSR2)*. Once the supply configuration has been configured, the ACTVOSRDY bit in the *PWR control status register 1 (PWR_SR1)* must be checked to guarantee valid voltage levels:
   a) As long as ACTVOSRDY indicates that voltage levels are invalid, the system is in Run* mode, write accesses to the RAMs are not permitted and VOS must not be changed.
   b) Once ACTVOSRDY indicates that voltage levels are valid, the system is in normal Run mode, write accesses to RAMs are allowed and VOS can be changed.

### Table 44. Supply configuration control (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Supply configuration</th>
<th>SDHILEVEL</th>
<th>SMPSEXTHP</th>
<th>SDEN</th>
<th>LDOEN</th>
<th>BYPASS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NA</td>
<td>Illegal</td>
<td>x</td>
<td>x</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>– Illegal combination, the default configuration is kept. (write data is ignored).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>–</td>
<td>– The cascade mode is prohibited and not detected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>x</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
VCORE directly supplied from the SMPS step-down converter

When VCORE is directly supplied from the SMPS step-down converter, the VCORE voltage first settles at VFBSMPS default level (1.36 V). Due to a too high supply compared to the VOS level, the ACTVOSRDY bit in the PWR control status register 1 (PWR_SR1) indicates invalid voltage levels. VCORE settles at VOS low and ACTVOSRDY indicates valid voltage levels only when the supply configuration has been programmed in PWR control register 2 (PWR_CSR2) (see Figure 22).

**Figure 22. Device startup with VCORE supplied directly from SMPS step-down converter**

1. In Run* mode, write operations to RAM are not allowed.
2. Write operations to RAM are allowed and VOS can be changed only when ACTVOSRDY is valid.
When exiting from Standby mode, the supply configuration is known by the system since the content of **PWR control register 2 (PWR_CSR2)** is retained. However the software must still wait for the **ACTVOSRDY** bit to be set in **PWR control status register 1 (PWR_SR1)** to indicate V\textsubscript{CORE} voltage levels are valid, before performing write accesses to RAM or changing VOS.

**V\textsubscript{CORE} supplied in Bypass mode (LDO and SMPS OFF)**

The devices that feature the SMPS can also be used in Bypass mode. When V\textsubscript{CORE} is supplied in Bypass mode (LDO and SMPS OFF), the V\textsubscript{CORE} voltage must first settles at a default level higher than 1.2 V. Due to the LDO default state after power-up (enabled by default), the external V\textsubscript{CORE} voltage must remain higher than 1.2 V until the LDO is disabled by software.

When the LDO is disabled, the external V\textsubscript{CORE} voltage can be adjusted according to the user application needs (refer to section **General operating conditions** of the datasheet for details on V\textsubscript{CORE} level versus the maximum operating frequency).

![Figure 23. Device startup with V\textsubscript{CORE} supplied in Bypass mode from external regulator](MSv53472V1)
6.4.2 Core domain

The $V_{\text{CORE}}$ core domain supply can be provided by the SMPS step-down converter, LDO voltage regulator or by an external supply ($V_{\text{CAP}}$). $V_{\text{CORE}}$ supplies all the digital circuitries except for the backup domain and the Standby circuitry. When a system reset occurs, the voltage regulator is enabled and supplies $V_{\text{CORE}}$. The SMPS step-down converter is also enabled to deliver 1.36 V. This allows the system to start up in any supply configurations (see Figure 21).

After a system reset, the software must configure the used supply configuration in $PWR$ control register 2 ($PWR\_CSR2$) register before changing VOS in the $PWR$ control status register 4 ($PWR\_CSR4$) or the RCC sys_ck frequency. The different system supply configurations are controlled as shown in Table 44.

Note: The SMPS step-down converter and the LDO are not available on all packages.

LDO voltage regulator

The embedded voltage regulator (LDO) requires external capacitors to be connected to VCAP pins.

The voltage regulator provides three different operating modes: High Power (HP), Low-power (LP) or Off. These modes are used depending on the system operating modes (Run, Stop and Standby).

- Run modes
  The LDO regulator is in Main mode and provides full power to the $V_{\text{CORE}}$ domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS low and VOS high) that are configured through the VOS bit in the $PWR$ control status register 4 ($PWR\_CSR4$). The VOS voltage scaling allows optimizing the power consumption when the system is clocked below the maximum frequency. By default VOS low is selected after system reset. VOS can be changed on-the-fly to adapt to the required system performance.

- Stop mode
  The voltage regulator supplies the $V_{\text{CORE}}$ domain to retain the content of registers and internal memories, and must be set in LP mode.
  In LP mode the regulator mode is selected through the SVOS bit in the $PWR$ control register 1 ($PWR\_CR1$). Due to a lower voltage level for SVOS low scaling, the Stop mode consumption can be further reduced.

- Standby mode
  The voltage regulator is OFF and the $V_{\text{CORE}}$ domains are powered down. The content of the registers and memories is lost except for the Standby circuitry and the backup domain.

For more details, refer to the voltage regulator section in the datasheets.
SMPS step-down converter regulator

The SMPS step-down converter requires an external coil to be connected between the dedicated VLXSMPS pin and, via a capacitor, to VSS.

The SMPS step-down converter can be used in internal supply mode or external supply mode. The internal supply mode is used to directly supply the V_CORE domain, while the external supply mode is used to generate an intermediate supply level (V_DDExtern at 1.8 V) which can supply the voltage regulator and optionally an external circuitry.

The SMPS step-down converter works in three different power modes: High Power (HP), Low-power (LP) or Off.

When the SMPS step-down converter is used in internal supply mode, the converter operating modes depend on the system modes (Run, Stop, Standby) and are configured through the associated VOS and SVOS levels:

- **Run mode**
  The SMPS step-down converter operates in HP mode and provides full power to the V_CORE domain (core, memories and digital peripherals). The regulator output voltage can be scaled by software to different voltage levels (VOS low and VOS high) that are configured through the VOS bit in the PWR control status register 4 (PWR_CSR4). The VOS voltage scaling allows optimizing the power consumption when the system is clocked below the maximum frequency. By default VOS low is selected after system reset. VOS can be changed on-the-fly to adapt to the required system performance.

- **Stop mode**
  The SMPS step-down converter supplies the V_CORE domain to retain the content of registers and internal memories, and must be set in LP mode. The voltage regulator mode is selected through the SVOS in the PWR control register 1 (PWR_CR1). In LP mode only SVOS low and SVOS high scaling are allowed. Due to a lower voltage level for these scaling, the Stop mode consumption can be further reduced.

- **Standby mode**
  The SMPS step-down converter is OFF and the V_CORE domains are powered down. The content of the registers and memories are lost except for the Standby circuitry and the backup domain.

When the SMPS step-down converter supplies an external circuitry by generating an intermediate voltage level, the converter is forced ON and operates in HP mode. The intermediate voltage level is selected through SDHILEVEL bits in the PWR control register 2 (PWR_CSR2). V_DDExtern is supplied at all times with full power whatever the system modes (Run, Stop, Standby).

**Note:** The SMPS step-down converter is not available on all packages, the SMPS supplies the voltage regulator and optionally an external circuitry. The LDO may not be used in some configuration.
6.4.3 PWR external supply

When \( V_{\text{CORE}} \) is supplied from an external source (Bypass mode), different operating modes can be used depending on the system operating modes (Run, Stop or Standby):
- **In Run modes**
  The external source supplies full power to the \( V_{\text{CORE}} \) domain (core, memories and digital peripherals). The external source output voltage is scalable through different voltage levels (VOS low and VOS high). The externally applied voltage level must be reflected in the VOS bit of PWR_CSR4 register. The RAMs must only be accessed for write operations when the external applied voltage level matches VOS settings.
- **In Stop mode**
  The external \( V_{\text{CORE}} \) supply should be maintained at the VOS level or at least over 0.95 V to ensure proper internal wakeup.
- **In Standby mode**
  The wake mechanism should be monitored externally. Refer to the datasheet for the proper \( V_{\text{CORE}} \) ramp time.

6.4.4 Backup domain

To retain the content of the backup domain (RTC, backup registers and backup RAM) when \( V_{\text{DD}} \) is turned off, VBAT pin can be connected to an optional voltage which is supplied from a battery or from another source.

The switching to VBAT is controlled by the power-down reset embedded in the Reset block that monitors the \( V_{\text{DD}} \) supply.

---

**Warning:** During \( t_{\text{RSTTEMPO}} \) (temporization at \( V_{\text{DD}} \) startup) or after a PDR is detected, the power switch between VBAT and \( V_{\text{DD}} \) remains connected to VBAT.

During the startup phase, if \( V_{\text{DD}} \) is established in less than \( t_{\text{RSTTEMPO}} \) (see the datasheet for the value of \( t_{\text{RSTTEMPO}} \)) and \( V_{\text{DD}} > V_{\text{BAT}} + 0.6 \) V, a current may be injected into VBAT through an internal diode connected between \( V_{\text{DD}} \) and the power switch (VBAT).

If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.
When the $V_{DD}$ supply is present, the backup domain is supplied from $V_{DD}$. This allows saving $V_{BAT}$ power supply battery life time.

If no external battery is used in the application, it is recommended to connect $V_{BAT}$ externally to $V_{DD}$, and add a a 100 nF external ceramic capacitor between $V_{BAT}$ and $V_{SS}$.

When the $V_{DD}$ supply is present and higher than the PDR threshold, the backup domain is supplied by $V_{DD}$ and the following functions are available:

- PC14 and PC15 can be used either as GPIO or as LSE pins.
- PC13 can be used either as GPIO or as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 pin assuming they have been configured by the RTC or the TAMPER registers (see Section 49.3.3: GPIOs controlled by the RTC and TAMP).
- PB9/TAMP_IN2, PE1/TAMP_IN3 when configured by the RTC as tamper pins.

*Note:* Since the switch only sinks a limited amount of current, the use of PC13 and PC15 GPIOs is restricted: only one I/O can be used as an output at a time. These I/Os must not be used as current sources (for example to drive an LED). Refer to the DS.

In $V_{BAT}$ mode, when the $V_{DD}$ supply is absent and a supply is present on $V_{BAT}$, the backup domain is supplied by $V_{BAT}$ and the following functions are available:

- PC14 and PC15 can be used as LSE pins only.
- PC13 can be used as RTC_OUT1, RTC_TS, TAMP_IN1, TAMP_OUT2 pin assuming they have been configured by the RTC or the TAMPER registers (see Section 49.3.3: GPIOs controlled by the RTC and TAMP).
- PB9/TAMP_IN2, PE1/TAMP_IN3 when configured by the RTC as tamper pins.

**Accessing the backup domain**

After reset, the backup domain (RTC registers and RTC backup registers) is protected against possible unwanted write accesses. To enable access to the backup domain, set the DBP bit in the **PWR control register 1 (PWR_CR1)**.

For more detail on RTC and backup RAM access, refer to Section 7: Reset and clock control (RCC).

**Backup RAM**

The backup domain includes 4 Kbytes of backup RAM accessible in 32-bit, 16-bit or 8-bit data mode. The backup RAM is supplied from the backup regulator in the backup domain. When the backup regulator is enabled through BREN bit in the **PWR control status register 1 (PWR_CSR1)**, the backup RAM content is retained even in Standby and/or $V_{BAT}$ mode (it can be considered as an internal EEPROM if $V_{BAT}$ is always present.)

The backup regulator can be ON or OFF depending whether the application needs the backup RAM function in Standby or $V_{BAT}$ modes.
After a tamper event, if the RTC flag is set, the backup RAM cannot be used until an erase is explicitly requested. When a tamper event is generated, reading the backup RAM returns 0x0, and any write different from 0 is not effective. The backup RAM can be erased:

- through the Flash memory interface when a regression from nvstate closed to nvstate open is requested. See also Section 4: System security.
  
or
- by performing a simple dummy write with zero as data to any address of the backup RAM (this action resets the whole backup RAM when it is performed after a tamper event).

**Figure 24. Backup domain**

### 6.4.5 V\textsubscript{BAT} battery charging

When \( V_{DD} \) is present, the external battery connected to \( V_{BAT} \) can be charged through an internal resistance.

\( V_{BAT} \) charging can be performed either through a 5 k\( \Omega \) resistor or through a 1.5 k\( \Omega \) resistor, depending on the VBRS bit value in *PWR control register 2 (PWR_CSR2)*.

The battery charging is enabled by setting the VBE bit in the *PWR control register 2 (PWR_CSR2)*. It is automatically disabled in \( V_{BAT} \) mode.
6.4.6 **Analog supply**

**Separate \( V_{DDA} \) analog supply**

The analog supply domain is powered by dedicated \( V_{DDA} \) and \( V_{SSA} \) pads that allow the supply to be filtered and shielded from noise on the PCB, thus improving ADC conversion accuracy:

- The analog supply voltage input is available on a separate \( V_{DDA} \) pin.
- An isolated supply ground connection is provided on \( V_{SSA} \) pin.
Analog reference voltage $V_{\text{REF+}}/V_{\text{REF-}}$

To achieve better accuracy low-voltage signals, the ADC also has a separate reference voltage, available on $V_{\text{REF+}}$ pin. The user can connect a separate external reference voltage on $V_{\text{REF+}}$.

The $V_{\text{REF+}}$ controls the highest voltage, represented by the full scale value, the lower voltage reference ($V_{\text{REF-}}$) being connected to VSSA.

When enabled by ENVR bit in the VREFBUF control and status register (see Section 29: Voltage reference buffer (VREFBUF)), $V_{\text{REF+}}$ is provided from the internal voltage reference buffer. The internal voltage reference buffer can also deliver a reference voltage to external components through $V_{\text{REF+}}/V_{\text{REF-}}$ pins.

When the internal voltage reference buffer is disabled by ENVR, $V_{\text{REF+}}$ is delivered by an independent external reference supply voltage.

**Note:** $V_{\text{REF+}}$ and $V_{\text{REF-}}$ pins are not available on all packages (in this case they are connected internally respectively to VDDA and VSSA). **Do not** enable the internal voltage reference buffer when an external power supply is applied to the $V_{\text{REF+}}$ pin.

### 6.4.7 USB regulator

The USB transceivers are supplied from a dedicated $V_{\text{DD33USB}}$ supply that can be provided either by the integrated USB regulator, or by an external USB supply.

When enabled by USBREGEN bit in the PWR control register 2 (PWR_CSR2), the $V_{\text{DD33USB}}$ is provided from the USB regulator, which is powered through the VDD50USB pin generally connect to the USB VBUS line. Before using $V_{\text{DD33USB}}$, check that it is available by monitoring USB33RDY bit in the PWR control register 2 (PWR_CSR2). The $V_{\text{DD33USB}}$ supply level detector must be enabled through USB33DEN bit in the PWR_CSR2 register.

When the USB regulator is disabled through USBREGEN bit, VDD33USB can be provided from an external supply. In this case VDD33USB and VDD50USB must be connected together. The VDD33USB supply level detector must be enabled through USB33DEN bit in the PWR_CSR2 register before using the USB transceivers.

For more information on the USB regulator (see Section 62: USB on-the-go high-speed (OTG_HS)).

**Figure 25. USB supply configurations**
6.5 Power supply supervision

Power supply level monitoring is available on the following supplies:

- \( V_{DD} (V_{DDSMPS}) \) can be monitored via POR/PDR (see Section 6.5.1), BOR (see Section 6.5.2) and PVD monitor (see Section 6.5.3)
- \( V_{DDA} \) can be monitored via AVD monitor (see Section 6.5.4)
- \( V_{BAT} \) can be monitored via VBAT threshold (see Section 6.5.5)
- \( V_{SW} \) can be monitored via rst_vsw, which keeps \( V_{SW} \) domain in Reset mode as long as the level is not OK.
- \( V_{BKP} \) can be monitored via a BRRDY bit in the \textit{PWR control status register 1 (PWR_CSR1)}.
- \( V_{FBSMPS} \) can be monitored via a SDEXTRDY bit in the \textit{PWR control register 2 (PWR_CSR2)}.
- \( V_{DD33USB} \) can be monitored via the USB33RDY bit in the \textit{PWR control register 2 (PWR_CSR2)}.  

6.5.1 Power-on reset (POR)/power-down reset (PDR)

The system has an integrated POR/PDR circuitry that ensures proper startup operation.

The system remains in Reset mode when \( V_{DD} \) is below a specified \( V_{POR} \) threshold, without the need for an external reset circuit. Once the \( V_{DD} \) supply level is above the \( V_{POR} \) threshold, the system is taken out of reset (see Figure 26). For more details concerning the power-on/power-down reset threshold, refer to the electrical characteristics section of the datasheets.

By default the PDR is enabled.

![Figure 26. Power-on reset/power-down reset waveform](image)

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.2 Brownout reset (BOR)

During power-on, the Brownout reset (BOR) keeps the system under reset until the $V_{DD}$ supply voltage reaches the specified $V_{BOR}$ threshold.

The $V_{BOR}$ threshold is configured through system option bytes. By default, BOR is OFF. The following programmable $V_{BOR}$ thresholds can be selected:

- BOR OFF ($V_{BOR0}$)
- BOR Level 1 ($V_{BOR1}$)
- BOR Level 2 ($V_{BOR2}$)
- BOR Level 3 ($V_{BOR3}$)

For more details on the brown-out reset thresholds, refer to the section “Electrical characteristics” of the product datasheets.

A system reset is generated when the BOR is enabled and $V_{DD}$ supply voltage drops below the selected $V_{BOR}$ threshold.

BOR can be disabled by programming the system option bytes. To disable the BOR function, $V_{DD}$ must have been higher than $V_{BOR0}$ to start the system option byte programming sequence. The power-down is then monitored by the PDR (see Section 6.5.1).

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.3 Programmable voltage detector (PVD)

The PVD can be used to monitor the \( V_{DD} \) power supply by comparing it to a threshold selected by the PLS[2:0] bits in the PWR control register 1 (PWR.CR1). The PVD can also be used to monitor a voltage level on the PVD_IN pin. In this case PVD_IN voltage is compared to the internal \( V_{REFINT} \) level.

The PVD is enabled by setting the PVDE bit in the PWR control register 1 (PWR.CR1).

A PVDO flag is available in the PWR control status register 1 (PWR_SR1) to indicate if \( V_{DD} \) or PVD_IN voltage is higher or lower than the PVD threshold. This event is internally connected to the EXTI and can generate an interrupt, assuming it has been enabled through the EXTI registers. The PVDO output interrupt can be generated when \( V_{DD} \) or PVD_IN voltage drops below the PVD threshold and/or when \( V_{DD} \) or PVD_IN voltage rises above the PVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could perform emergency shutdown tasks.

**Figure 28. PVD thresholds**

![PVD thresholds diagram](MSv40342V2)

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.4 **Analog voltage detector (AVD)**

The AVD can be used to monitor the $V_{DDA}$ supply by comparing it to a threshold selected by the ALS[1:0] bits in the *PWR control register 1 (PWR_CR1)*.

The AVD is enabled by setting the AVDEN bit in the *PWR control register 1 (PWR_CR1)*.

An AVDO flag is available in the *PWR control status register 1 (PWR_SR1)* to indicate whether $V_{DDA}$ is higher or lower than the AVD threshold. This event is internally connected to the EXTI and can generate an interrupt if enabled through the EXTI registers. The AVDO interrupt can be generated when $V_{DDA}$ drops below the AVD threshold and/or when $V_{DDA}$ rises above the AVD threshold depending on EXTI rising/falling edge configuration. As an example the service routine could indicate when the $V_{DDA}$ supply drops below a minimum level.

**Figure 29. AVD thresholds**

1. For thresholds and hysteresis values, refer to the datasheets.
### 6.5.5 Battery voltage thresholds

The battery voltage supply monitors the backup domain $V_{SW}$ level. $V_{SW}$ is monitored by comparing it with two threshold levels: $V_{BAT_{high}}$ and $V_{BAT_{low}}$. $VBATH$ and $VBATL$ flags in the PWR control status register 1 ($PWR_{CSR1}$) indicate if $V_{SW}$ is higher or lower than the threshold.

The $V_{BAT}$ supply monitoring can be enabled/disabled via MONEN bit in the PWR control status register 1 ($PWR_{CSR1}$). When it is enabled, the battery voltage thresholds increase power consumption. As an example the $V_{SW}$ levels monitoring could be used to trigger a tamper event for an over or under voltage of the RTC power supply domain (available in VBAT mode).

$VBATH$ and $VBATL$ are connected to RTC tamper signals (see Section 49: Real-time clock (RTC)).

**Note:** Battery voltage monitoring is only available when the backup regulator is enabled (BREN bit set in the PWR control status register 1 ($PWR_{CSR1}$)).

When the device does not operate in VBAT mode, the battery voltage monitoring checks $V_{DD}$ level. When $V_{DD}$ is available, $V_{SW}$ is connected to $V_{DD}$ through the internal power switch (see Section 6.4.4: Backup domain).

**Figure 30. VBAT thresholds**

![VBAT thresholds diagram](MSv40344V1)

1. For thresholds and hysteresis values, refer to the datasheets.
### 6.5.6 Temperature thresholds

The junction temperature can be monitored by comparing it with two threshold levels, \( \text{TEMP}_{\text{high}} \) and \( \text{TEMP}_{\text{low}} \). TEMPH and TEMPL flags, in the PWR control status register 1 (PWR_CSR1), indicate whether the device temperature is higher or lower than the threshold. The temperature monitoring can be enabled/disabled via MONEN bit in the PWR control status register 1 (PWR_CSR1). When enabled, the temperature thresholds increase power consumption. As an example the levels could be used to trigger a routine to perform temperature control tasks.

The temperature thresholds are available only when the backup regulator is enabled (BREN bit set in the PWR_CSR1 register).

TEMPH and TEMPL wakeup interrupts are available on the RTC tamper signals (see Section 49: Real-time clock (RTC)).

![Figure 31. Temperature thresholds](image)

1. For thresholds and hysteresis values, refer to the datasheets.
6.5.7 \textit{V} \textsubscript{CORE} maximum voltage level detector

\( \text{V}_{\text{CORE}} \) is protected against too high voltages in the direct SMPS step-down converter supply configuration. \( \text{V}_{\text{CORE}} \) overvoltage protection is enabled at startup by hardware once the SMPS step-down converter configuration has been programmed into \textit{PWR control register 2 (PWR_CSR2)}:

- \( \text{V}_{\text{CORE}} \) voltage level stays within range:
  - ACTVOSRDY bit in the \textit{PWR control status register 1 (PWR_SR1)} indicate valid voltage levels.
  - The system operates normally and \( \text{V}_{\text{CORE}} \) overvoltage protection is disabled.
- \( \text{V}_{\text{CORE}} \) overvoltage (due to a wrongly programmed SMPS step-down converter configuration):
  - The hardware forces the SMPS step-down converter voltage level to 1.0 V.
  - The ACTVOSRDY goes on indicating invalid voltage levels. In this case the software must be corrected and re-loaded to program a correct SMPS step-down converter configuration that matches the application supply connections. The system must then be power cycled.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{msv55525v1.png}
\caption{\textit{V} \textsubscript{CORE} overvoltage protection}
\end{figure}
6.6 Power management

The power management block controls the $V_{\text{CORE}}$ supply in accordance with the system operation modes (see Section 6.6.1).

The device power domains can operate in one of the following operating modes:

- Run (power ON, clock ON)
- Sleep (power ON, core clock stopped, peripherals keep running)
- Stop (power ON, clock OFF)
- Standby (Power OFF, clock OFF).

The $V_{\text{CORE}}$ supply level follows the system operating mode (Run, Stop, Standby).

The following voltage scaling features allow controlling the power with respect to the required system performance (see Section 6.6.2: Voltage scaling):

- To obtain a given system performance, the corresponding voltage scaling must be set in accordance with the system clock frequency. To do this, configure the VOS bits to the Run mode voltage scaling.
- To obtain the best trade-off between power consumption and exit-from-Stop mode latency, configure the SVOS bit to Stop mode voltage scaling.

6.6.1 Operating modes

Several system operating modes are available to tune the system according to the performance required, that is, when the CPU does not need to execute code and are waiting for an external event. It is up to the user to select the operating mode that gives the best compromise between low power consumption, short startup time and available wakeup sources.

The operating modes allow controlling the clock distribution to the different system blocks and powering them.

In Run mode, power consumption can be reduced by one of the following means:

- Lowering the system performance by slowing down the system clocks and reducing the $V_{\text{CORE}}$ supply level through VOS voltage scaling bit.
- Gating the clocks to the APBx and AHBx peripherals when they are not used, through PERxEN bits.
The VCORE domain is supplied from a single voltage regulator supporting voltage scaling with the following features:

- **Run mode voltage scaling**
  - VOS low
  - VOS high
- **Stop mode voltage scaling**
  - SVOS low
  - SVOS high

For more details on voltage scaling values, refer to the product datasheets.

After reset, the system starts on the lowest Run mode voltage scaling (VOS low). The voltage scaling can then be changed on-the-fly by software by programming the VOS bit in the `PWR control status register 4 (PWR_CSR4)` according to the required system performance. When exiting from Stop mode or Standby mode, the Run mode voltage scaling is reset to the default VOS low value.

Before entering Stop mode, the software can preselect the SVOS level in `PWR control register 1 (PWR_CR1)`. The Stop mode voltage scaling for SVOS low and SVOS high also sets the voltage regulator in Low-power (LP) mode to further reduce power consumption.

In Standby mode the VCORE supply is switched off.
6.6.3 Power management examples

- Figure 34 shows $V_{\text{CORE}}$ voltage scaling behavior in Run mode.
- Figure 35 shows $V_{\text{CORE}}$ voltage scaling behavior in Stop mode.
- Figure 36 shows $V_{\text{CORE}}$ voltage regulator and voltage scaling behavior in Standby mode.

Example of $V_{\text{CORE}}$ voltage scaling behavior in Run mode

Figure 34 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS low.
2. The system performance is then increased to a high-speed clock from the PLL with voltage scaling VOS high. To do this:
   a) Program the voltage scaling to VOS high.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. The next step is to reduce the system performance to HSI clock with voltage scaling VOS low. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS low.
4. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS high.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly, otherwise the system might be unreliable.

**Figure 34. Dynamic voltage scaling in Run mode**

1. The status of the register bits at each step is shown in blue.

**Example of $V_{\text{CORE}}$ voltage scaling behavior in Stop mode**

*Figure 35* illustrates the following system operation sequence example:
1. The system is running from the PLL in high-performance mode (VOS high voltage scaling).
2. First the CPU subsystem deallocates all the peripheral, in a second step, the CPU subsystem enters Stop mode and the system enters Stop mode. The system clock is stopped and the hardware lowers the voltage scaling to the software preselected SVOS low.
3. The CPU subsystem is then woken up. The system exits from Stop mode and the CPU subsystem exits from Stop mode. The hardware always sets the voltage scaling to VOS low after exiting from Stop mode and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock is enabled and switch in run mode under HSI clock.

4. The system performance could then be increased. To do this:
   a) The software first sets the voltage scaling to VOS high.
   b) Once the $V_{\text{CORE}}$ supply has reached the required level indicated by VOSRDY, the clock frequency can be increased by enabling the PLL.
   c) Once the PLL is locked, the system clock can be switched.

**Figure 35. Dynamic voltage scaling behavior in Stop mode**

1. The status of the register bits at each step is shown in blue.

**Example of $V_{\text{CORE}}$ voltage regulator and voltage scaling behavior in Standby mode**

**Figure 36** illustrates the following system operation sequence example:

1. The system is running from the PLL in high-performance mode (VOS high voltage scaling).
2. The CPU subsystem deallocates all the peripherals and the power is switched off. The system enter in standby mode.
3. The system is then woken up by a wakeup source. The system exits from Standby mode. The hardware sets the voltage scaling to the default VOS low and waits for the requested supply level to be reached before enabling the default HSI oscillator. Once
the HSI clock is stable, the system clock is enabled. The software must then check the
ACTVOSRDKY is valid before changing the system performance.

4. In a next step, increase the system performance. To do this:
   a) The software first increases the voltage scaling to VOS high
   b) Before enabling the PLL, it waits for the requested supply level to be reached by
      monitoring VOSRDY bit.
   c) Once the PLL is locked, the system clock can be switched.

Figure 36. Dynamic voltage scaling from Standby mode

1. The status of the register bits at each step is shown in blue.

6.7 Low-power modes

Several low-power modes are available to save power when the CPU does not need to
execute code (i.e. when waiting for an external event). It is up to the user application to
select the mode that gives the best compromise between low power consumption, short
startup time and available wakeup sources:

- Slowing down system clocks (see Section 7.5.6: System clock (sys_clk))
- Controlling individual peripheral clocks (see Section 7.5.12: Peripheral clock gating
  control)
- Low-power modes
  - Sleep (CPU clock stopped and still in RUN mode)
  - Stop (System clock stopped)
  - Standby (System powered down)
6.7.1 **Slowing down system clocks**

In Run mode, the speed of the system clock sys_ck can be reduced. For more details refer to Section 7.5.6: System clock (sys_ck).

6.7.2 **Controlling peripheral clocks**

In Run mode, the HCLKx and PCLKx for individual peripherals can be stopped by configuring at any time PERxEN bit in RCC_xxxxENR to reduce power consumption.

To reduce power consumption in CSleep mode, the individual peripheral clocks can be disabled by configuring PERxLPEN bit in RCC_xxxxLPENR. For the peripherals still receiving a clock in Sleep mode, their clock can be slowed down before entering Sleep mode.

Example: for APB2 peripherals: the RCC registers is named: RCC_APB2ENR, this register contains SPI4EN, SAI1EN, and so on.

6.7.3 **Entering low-power modes**

CPU subsystem Sleep and Stop low-power modes are entered by the MCU when executing the WFI (Wait For Interrupt) or WFE (Wait for Event) instructions, or when the SLEEPONEXIT bit in the Cortex®-M System Control register is set on Return from ISR.

The system can enter Stop or Standby low-power mode when all EXTI wakeup sources are cleared.

6.7.4 **Exiting from low-power modes**

The CPU subsystem exits from Sleep mode through any interrupt or event depending on how the low-power mode was entered:

- If the WFI instruction or Return from ISR was used to enter to low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the system.
- If the WFE instruction is used to enter to low-power mode, the CPU exits from low-power mode as soon as an event occurs. The wakeup event can be generated either by:
  - An NVIC IRQ interrupt.
    
    **When SEVONPEND = 0** in the Cortex®-M7 System Control register, the interrupt must be enabled in the peripheral control register and in the NVIC.
    
    When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit in the NVIC interrupt clear pending register have to be cleared. Only NVIC interrupts with sufficient priority wakes up and interrupts the MCU.

    **When SEVONPEND = 1** in the Cortex®-M7 System Control register, the interrupt must be enabled in the peripheral control register and optionally in the NVIC.
    
    When the MCU resumes from WFE, the peripheral interrupt pending bit and, when
enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

All NVIC interrupts wakeup the MCU, even the disabled ones.
Only enabled NVIC interrupts with sufficient priority wakeup and interrupt the MCU.

– An event

An EXTI line must be configured in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI peripheral interrupt pending bit or the NVIC IRQ channel pending bit as the pending bits corresponding to the event line is not set. It might be necessary to clear the interrupt flag in the peripheral.

The CPU subsystem exits from Stop modes by enabling an EXTI interrupt or event depending on how the low-power mode was entered (see above).

The CPU subsystem exits from Standby mode by enabling an EXTI interrupt or event. Program execution restarts from CPU local reset (such as a reset vector fetched from the system configuration boot and security block (SBS).

The CPU subsystem exits from Standby mode by enabling an external reset (NRST pin), an IWDG reset, a rising edge on one of the enabled WKUPx pins or a RTC event. Program execution restarts in the same way as after a system reset (such as boot pin sampling, option bytes loading or reset vector fetched).
6.7.5 Sleep mode

The Sleep mode applies only to the CPU subsystem. In Sleep mode, the CPU clock is stopped. The CPU subsystem peripheral clocks operate according to the values of PERxLPEN bits in RCC_xxxxENR.

Entering Sleep mode

The Sleep mode is entered according to Section 6.7.3: Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M System Control register is cleared.

Refer to Table 46 for details on how to enter to Sleep mode.

Exiting from Sleep mode

The Sleep mode is exited according to Section 6.7.4: Exiting from low-power modes.

Refer to Table 46 for more details on how to exit from Sleep mode.

<table>
<thead>
<tr>
<th>Sleep mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode entry</td>
<td>WFI (Wait for Interrupt) or WFE (Wait for Event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0 (refer to the Cortex®-M System Control register.)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared.</td>
</tr>
<tr>
<td></td>
<td>On return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 0 and</td>
</tr>
<tr>
<td></td>
<td>– SLEEPONEXIT = 1 (refer to the Cortex®-M System Control register.)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared.</td>
</tr>
</tbody>
</table>

| Mode exit | If WFI or return from ISR was used for entry: |
|           | – Any Interrupt enabled in NVIC: Refer to Table 142: NVIC |
|           | If WFE was used for entry and SEVONPEND = 0: |
|           | – Any event: Refer to Section 20.5.3: EXTI CPU wakeup procedure |
|           | If WFE was used for entry and SEVONPEND = 1: |
|           | – Any Interrupt even when disabled in NVIC: refer to Table 142: NVIC or any event: refer to Section 20.5.3: EXTI CPU wakeup procedure |

| Wakeup latency | None |

6.7.6 Stop mode

The Stop mode applies to the MCU. In Stop mode, the CPU clock is stopped. All CPU subsystem peripheral clocks are stopped too and.

In Stop mode, CPU subsystem peripherals having a kernel clock request can still request their kernel clock.

The Flash memory can enter low-power Stop mode when it is enabled through FLPS in PWR_CR1 register. This allows a trade-off between domain Stop restart time and low power consumption.

The HSI or CSI can remain enabled in system Stop mode (HSIKERON and CSIKERON set in RCC_CR register). After exiting Stop mode, the clock is quickly available as kernel clock.
for peripherals. Other system oscillator sources are stopped in Stop mode and require a starting time after exiting Stop mode.

In Stop mode and SVOS high, peripherals using the LSI or LSE clock and peripherals having a kernel clock request are still able to operate.

In system Stop mode, the following features can be selected to remain active by programming individual control bits:

- **Independent watchdog (IWDG)**
  The IWDG is started by writing to its key register or by hardware option. Once started it cannot be stopped except by a reset (see Section 48.3 in Section 48: Independent watchdog (IWDG)).

- **Real-time clock (RTC)**
  This is configured via the RTCEN bit in the RCC Backup domain control register (RCC_BDCR).

- **Internal RC oscillator (LSI RC)**
  This is configured via the LSION bit in the RCC clock control and status register (RCC_CSR).

- **External 32.768 kHz oscillator (LSE OSC)**
  This is configured via the LSEON bit in the RCC Backup domain control register (RCC_BDCR).

- **Peripherals capable of running on the LSI or LSE clock.**

- **Peripherals having a kernel clock request.**

- **Internal RC oscillators (HSI and CSI)**
  This is configured via the HSIKERON and CSIKERON bits in the RCC source control register (RCC_CR).

- **The ADC can also consume power during Stop mode, unless they are disabled before entering this mode. To disable them, the ADON bit in the ADC_CR2 register must be written to 0.**

The selected SVOS low levels add an additional startup delay when exiting from system Stop mode (see Table 47).

<table>
<thead>
<tr>
<th>SVOS</th>
<th>Stop mode Voltage regulator operation</th>
<th>Wake-up Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVOS high</td>
<td>LP</td>
<td>Voltage Regulator wakeup time from SVOS high LP mode to VOS low</td>
</tr>
<tr>
<td>SVOS low</td>
<td>LP</td>
<td>Voltage Regulator wakeup time from SVOS low LP mode to VOS low</td>
</tr>
</tbody>
</table>
**Entering Stop mode**

The Stop mode is entered according to Section 6.7.3: Entering low-power modes, when the SLEEPDEEP bit in the Cortex®-M System Control register is set.

If Flash memory programming is ongoing, the Stop mode entry is delayed until the memory access is finished.

If an access to the domain bus matrix is ongoing, the Stop mode entry is delayed until the domain bus matrix access is complete.

See Table 48 below for details on how to enter to Stop mode.

<table>
<thead>
<tr>
<th>Stop mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mode entry</strong></td>
<td>WiFi (Wait for Interrupt) or WFE (Wait for Event) while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 1 (refer to the Cortex®-M System Control register.)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared.</td>
</tr>
<tr>
<td></td>
<td>– All CPU EXTI Wakeup sources are cleared.</td>
</tr>
<tr>
<td></td>
<td>On return from ISR while:</td>
</tr>
<tr>
<td></td>
<td>– SLEEPDEEP = 1 and</td>
</tr>
<tr>
<td></td>
<td>– SLEEPONEXIT = 1 (refer to the Cortex®-M System Control register.)</td>
</tr>
<tr>
<td></td>
<td>– CPU NVIC interrupts and events cleared.</td>
</tr>
<tr>
<td></td>
<td>– All CPU EXTI Wakeup sources are cleared.</td>
</tr>
<tr>
<td><strong>Mode exit</strong></td>
<td>If WFI or return from ISR was used for entry:</td>
</tr>
<tr>
<td></td>
<td>– EXTI Interrupt enabled in NVIC: refer to Table 142: NVIC, for peripheral which are not stopped or powered down.</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 0:</td>
</tr>
<tr>
<td></td>
<td>– EXTI event: refer to Section 20.5.3: EXTI CPU wakeup procedure, for peripheral which are not stopped or powered down.</td>
</tr>
<tr>
<td></td>
<td>If WFE was used for entry and SEVONPEND = 1:</td>
</tr>
<tr>
<td></td>
<td>– EXTI Interrupt even when disabled in NVIC: refer to Table 142: NVIC or EXTI event: refer to Section 20.5.3: EXTI CPU wakeup procedure, for peripheral which are not stopped or powered down.</td>
</tr>
<tr>
<td><strong>Wakeup latency</strong></td>
<td>EXTI and RCC wakeup synchronization (see Section 7.4.11: Power-on and wakeup sequences)</td>
</tr>
</tbody>
</table>
To allow peripherals having a kernel clock request to operate in Stop mode, the system must use SVOS high.

**Note:** *Use a DSB instruction to ensure that outstanding memory transactions complete before entering stop mode.*

**Exiting from Stop mode**

The Stop mode is exited according to *Section 6.7.4: Exiting from low-power modes.*

Refer to *Table 48: Stop mode* for more details on how to exit from Stop mode.

When exiting from Stop mode, the system clock, bus matrix clocks and voltage scaling are reset.

STOPF status flag in the *PWR CPU control register 3 (PWR_CSR3)* indicates that the system has exited from Stop mode.

**I/O states in Stop mode**

I/O pin configuration remain unchanged in Stop mode.

### 6.7.7 Standby mode

The Standby mode allows achieving the lowest power consumption. Like Stop mode, it is based on CPU subsystem Stop mode. However the \( V_{CORE} \) supply regulator is powered off.

When the system enters in Standby mode, the voltage regulator is disabled. The complete \( V_{CORE} \) domain is consequently powered off. The PLLs, HSI oscillator, CSI oscillator, HSI48 and the HSE oscillator are also switched off. SRAM and register contents are lost except for backup domain registers (RTC registers, RTC backup register and backup RAM), and Standby circuitry (see *Section 6.4.4: Backup domain*).

In system Standby mode, the following features can be selected by programming individual control bits:

- **Independent watchdog (IWDG)**
  
  The IWDG is started by programming its Key register or by hardware option. Once started, it cannot be stopped except by a reset (see *Section 48.3 in Section 48: Independent watchdog (IWDG).*

- **Real-time clock (RTC)**
  
  This is configured via the RTCEN bit in the backup domain control register (RCC_BDCR).

- **Internal RC oscillator (LSI RC)**
  
  This is configured by the LSION bit in the Control/status register (RCC_CSR).

- **External 32.768 kHz oscillator (LSE OSC)**
  
  This is configured by the LSEON bit in the backup domain control register (RCC_BDCR).
Entering Standby mode

The Standby mode is entered according to Section 6.7.3: Entering low-power modes, when the PDDS bits in the PWR CPU control register 3 (PWR_CSR3) is set to one.

If Flash memory programming is ongoing, the Standby mode entry is delayed until the memory access is finished.

Refer to Table 50 for more details on how to enter to Standby mode.

Exiting from Standby mode

The Standby mode is exited according to Section 6.7.4: Exiting from low-power modes.

The system exits from Standby mode when an external Reset (NRST pin), an IWDG Reset, a WKUP pin event, a RTC alarm, a tamper event, or a time stamp event is detected. All registers are reset after waking up from Standby except for power control and status registers (PWR control register 2 (PWR_CSR2), PWR CPU control register 3 (PWR_CSR3)), SBF bit in the PWR CPU control register 3 (PWR_CSR3), PWR wakeup flag register (PWR_WKUPFR), and PWR wakeup enable and polarity register (PWR_WKUPEPR).

After waking up from Standby mode, the program execution restarts in the same way as after a system reset (boot option sampling, boot vector reset fetched, etc.). The SBF status flags in the PWR CPU control register 3 (PWR_CSR3) registers indicate from which mode the system has exited (refer to Table 50).

Table 49. Standby and Stop flags

<table>
<thead>
<tr>
<th>SBF</th>
<th>STOPF</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>System has been in Stop</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>System has been in Standby</td>
</tr>
</tbody>
</table>

Table 50. Standby mode

<table>
<thead>
<tr>
<th>Standby mode</th>
<th>Description</th>
</tr>
</thead>
</table>
| Mode entry   | – The CPU subsystem is in Stop mode, and there is no active EXTI Wakeup source.  
                 – PDDS bit for select Standby.  
                 – All WKUPF bits in the Power Control/Status register (PWR_WKUPFR) are cleared. |
| Mode exit    | – WKUP pins rising or falling edge, RTC alarm (Alarm A and Alarm B), RTC wakeup, tamper event, time stamp event, external reset in NRST pin, IWDG reset. |
| Wakeup latency | System reset phase (see Section 7.4.2: The system and application resets (sys_rst and nreset)) |
I/O states in Standby mode

In Standby mode, all I/O pins are high impedance without pull, except for:
- Reset pad (still available)
- RTC_AF1 pin if configured for tamper, time stamp, RTC Alarm out, or RTC clock calibration out
- WKUP pins (if enabled). The WKUP pin pull configuration can be defined through WKUPPUPD register bits in the PWR wakeup enable and polarity register (PWR_WKUPEPR)
- I2C pin-pull configuration if enabled through the register PWR_APCR.

6.7.8 Monitoring low-power modes

The devices feature state monitoring pins to monitor the state transition to low-power mode (refer to Table 51 for the list of pins and their description). The GPIO pin corresponding to each monitoring signal has to be programmed in alternate function mode.

This feature is not available in Standby mode since these I/O pins are switched to high impedance.

<table>
<thead>
<tr>
<th>Power state monitoring pins</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PWR_CSLEEP (CSLEEP)</td>
<td>Sleeping CPU state</td>
</tr>
<tr>
<td>PWR_CSTOP (CDSLEEP)</td>
<td>Deepsleep CPU state</td>
</tr>
</tbody>
</table>

The values of the monitoring pins reflect the state of the CPU and domains. Refer to Table 52 for the GPIO state depending on CPU and domain state.

<table>
<thead>
<tr>
<th>CPU</th>
<th>CPU power state</th>
<th>Power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSLEEP</td>
<td>CDSLEEP</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Run mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>CPU in Sleep mode</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>CPU in Run mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>CPU in Deepsleep mode</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>(.1) Standby mode</td>
</tr>
</tbody>
</table>

1. The CPU is powered off.
6.8 **PWR registers**

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

### 6.8.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 0001

| Bit 31-16 | Reserved, must be kept at reset value. |
| Bit 15-14 | ALS[1:0]: Analog voltage detector level selection |
| | These bits select the voltage threshold detected by the AVD. |
| | 00: AVD level 1 |
| | 01: AVD level 2 |
| | 10: AVD level 3 |
| | 11: AVD level 4 |
| Bit 13 | AVDEN: Peripheral voltage monitor on VDDA enable |
| | 0: Peripheral voltage monitor on VDDA disabled |
| | 1: Peripheral voltage monitor on VDDA enabled |
| Bit 12 | AVDREADY: analog voltage ready |
| | This bit is only used when the analog switch boost needs to be enabled (see BOOSTE bit). |
| | It must be set by software when the expected VDDA analog supply level is available. |
| | The correct analog supply level is indicated by the AVDO bit (PWR_CSR1 register) after setting the AVDEN bit and selecting the supply level to be monitored (ALS bits). |
| | 0: peripheral analog voltage VDDA not ready (default) |
| | 1: peripheral analog voltage VDDA ready |
| Bit 11 | BOOSTE: analog switch VBoost control |
| | This bit enables the booster to guarantee the analog switch AC performance when the VDD supply voltage is below 2.7 V (reduction of the total harmonic distortion to have the same switch performance over the full supply voltage range) |
| | The VDD supply voltage can be monitored through the PVD and the PLS bits. |
| | 0: booster disabled (default) |
| | 1: booster enabled if analog voltage ready (AVD_READY = 1) |
| Bit 10 | Reserved, must be kept at reset value. |
Bit 9  **FLPS**: Flash low-power mode in Stop mode
This bit allows to obtain the best trade-off between low-power consumption and restart time when exiting from Stop mode.
When it is set, the Flash memory enters low-power mode when device is in Stop mode.
0: Flash memory remains in normal mode when device enters Stop (quick restart time).
1: Flash memory enters low-power mode when device enters Stop mode (low-power consumption).

Bit 8  **DBP**: Disable backup domain write protection
In reset state, the RCC_BDCR register, the RTC registers (including the backup registers), BREN and MOEN bits in the PWR_CSR1 register, are protected against parasitic write access.
This bit must be set to enable write access to these registers.
0: Access to RTC, RTC backup registers and backup SRAM disabled
1: Access to RTC, RTC backup registers and backup SRAM enabled

Bits 7:5  **PLS[2:0]**: Programmable voltage detector level selection
These bits select the voltage threshold detected by the PVD.
000: PVD level 1
001: PVD level 2
010: PVD level 3
011: PVD level 4
100: PVD level 5
101: PVD level 6
110: PVD level 7
111: External voltage level on PVD_IN pin
111: External voltage level on PVD_IN pin, compared to internal VREFINT level.

*Note: Refer to Section "Electrical characteristics" of the product datasheet for more details.*

Bit 4  **PVDE**: Programmable voltage detector enable
0: Programmable voltage detector disabled.
1: Programmable voltage detector enabled

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **SVOS**: System Stop mode voltage scaling selection.
0: SVOS low
1: SVOS high (default)
### 6.8.2 PWR control status register 1 (PWR_SR1)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 13    | AVDO: Analog voltage detector output on VDDA  
This bit is set and cleared by hardware. It is valid only if AVD on VDDA is enabled by the AVDEN bit.  
0: VDDA is equal or higher than the AVD threshold selected with the ALS[1:0] bits.  
1: VDDA is lower than the AVD threshold selected with the ALS[1:0] bits  
**Note:** Since the AVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the AVDEN bit is set |
| Bit 12:5  | Reserved, must be kept at reset value. |
| Bit 4     | PVDO: Programmable voltage detect output  
This bit is set and cleared by hardware. It is valid only if the PVD has been enabled by the PVDE bit.  
0: VDD or PVD_IN voltage is equal or higher than the PVD threshold selected through the PLS[2:0] bits.  
1: VDD or PVD_IN voltage is lower than the PVD threshold selected through the PLS[2:0] bits.  
**Note:** Since the PVD is disabled in Standby mode, this bit is equal to 0 after Standby or reset until the PVDE bit is set |
| Bit 3:2   | Reserved, must be kept at reset value. |
| Bit 1     | ACTVOSRDY: Voltage levels ready bit for currently used ACTVOS and SDHILEVEL  
This bit is set to 1 by hardware when the voltage regulator and the SMPS step-down converter are both disabled and Bypass mode is selected in PWR control register 2 (PWR_CSR2).  
0: Voltage level invalid, above or below current ACTVOS and SDHILEVEL selected levels.  
1: Voltage level valid, at current ACTVOS and SDHILEVEL selected levels. |
| Bit 0     | ACTVOS: VOS currently applied for VCORE voltage scaling selection.  
These bit reflect the last VOS value applied to the PMU.  
0: VOS low level  
1: VOS high level |
6.8.3 PWR control status register 1 (PWR_CSR1)

Address offset: 0x008
Reset value: 0x0000 0000

This register is not reset by wakeup from Standby mode, RESET signal and V_DD POR. It is only reset by V_SW POR and VSWRST reset.

This register must not be accessed when VSWRST bit in the RCC_BDCR register resets the V_SW domain.

After reset, the PWR_CSR1 register is write-protected. Prior to modifying its content, the DBP bit in the PWR_CR1 register must be set to disable the write protection.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TEMPH</td>
<td>TEMPL</td>
<td>VBATL</td>
<td>VBATH</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 TEMPH: Temperature level monitoring versus high threshold
0: Temperature below high threshold level.
1: Temperature equal or above high threshold level.

Bit 22 TEMPL: Temperature level monitoring versus low threshold
0: Temperature above low threshold level.
1: Temperature equal or below low threshold level.

Bit 21 VBATL: V_BAT level monitoring versus low threshold
0: V_BAT level above low threshold level.
1: V_BAT level equal or below low threshold level.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 BRRDY: Backup regulator ready
0: Backup regulator not ready.
1: Backup regulator ready.

Bits 15:5 Reserved, must be kept at reset value.
Bit 4 **MONEN**: V\textsubscript{BAT} and temperature monitoring enable

- When set, the V\textsubscript{BAT} supply and temperature monitoring is enabled.
- 0: V\textsubscript{BAT} and temperature monitoring disabled.
- 1: V\textsubscript{BAT} and temperature monitoring enabled.

*Note:* V\textsubscript{BAT} and temperature monitoring are only available when the backup regulator is enabled (BREN bit set to 1).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **BREN**: Backup regulator enable

- When set, the backup regulator (used to maintain the backup RAM content in Standby and V\textsubscript{BAT} modes) is enabled.
- If BREN is reset, the backup regulator is switched off. The backup RAM can still be used in Run and Stop modes. However, its content is lost in Standby and V\textsubscript{BAT} modes.
- If BREN is set, the application must wait till the backup regulator ready flag (BRRDY) is set to indicate that the data written into the SRAMs is maintained in Standby and V\textsubscript{BAT} modes.
- 0: Backup regulator disabled.
- 1: Backup regulator enabled.
6.8.4 PWR control register 2 (PWR_CSR2)

Address offset: 0x00C
Reset value: 0x0000 0006

This register is reset only by a power-on reset (POR). It is not reset by a wakeup from Standby mode or RESET signal.

The lower byte of this register is written once after POR and must be written before entering low power mode, changing VOS level.

The lower byte of this register should be written as soon as possible after device start and the regulator state should be verified by reading ACTVOSRDY before entering the low power mode or changing VOS level.

No limitation applies to the upper bytes.

Programming data corresponding to an invalid combination of SDHILEVEL, SMPSEXTHP, SDEN, LDOEN and BYPASS bits (see Table 44: Supply configuration control) is ignored; data is not written because the written-once mechanism locks the register and any further write access is ignored.

The default supply configuration is kept, and the ACTVOSRDY bit in the PWR control status register 1 (PWR_SR1) continues to indicate invalid voltage levels. The system must be power cycled before writing a new value.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **USBHSRGEN**: USB HS regulator enable.
0: USB HS PHY regulator disabled (default)
1: USB HS PHY regulator enabled

*Note:* This bit should only be set if USB HS is used

Bit 26 **USB33RDY**: USB supply ready.
0: USB33 supply not ready
1: USB33 supply ready.

Bit 25 **USBRGEN**: USB regulator enable.
0: USB regulator disabled
1: USB regulator enabled.

Bit 24 **USB33DEN**: VDD33USB voltage level detector enable
0: VDD33USB voltage level detector disabled
1: VDD33USB voltage level detector enabled.

*Note:* This bit should be set if USB HS, FS, or a GPIO on PMx is used

Bits 23:17 Reserved, must be kept at reset value.
Bit 16 **SDEXT RDY**: SMPS step-down converter external supply ready
   This bit is set by hardware to indicate that the external supply from the SMPS step-down converter is ready.
   0: External supply not ready.
   1: External supply ready.

Bit 15 **EN_XSPIM2**: this bit allows the SW to enable the XSPI interface, when available. The XSPIM_P2 supply must be stable prior to setting this bit. It should also be set when FMC is used.
   0: XSPIM2 interface is not enabled
   1: XSPIM2 interface is enabled
   **CAUTION**: Not respecting this condition could damage the device.

Bit 14 **EN_XSPIM1**: this bit allow the SW to enable the XSPI interface. The XSPIM_P1 supply must be stable prior to setting this bit.
   0: XSPIM1 interface is not enabled
   1: XSPIM1 interface is enabled
   **CAUTION**: not respecting this condition could damage the device.

Bits 13:12 **XSPICAP2[1:0]**: XSPI port 2 capacitor control bits
   00: XSPI Capacitor OFF (default)
   01: XSPI Capacitor set to 1/2
   10: XSPI Capacitor set to 1/2
   11: XSPI Capacitor set to full capacitance
   see the product datasheet for more details

Bits 11:10 **XSPICAP1[1:0]**: XSPI port 1 capacitor control bits
   00: XSPI Capacitor OFF (default)
   01: XSPI Capacitor set to 1/2
   10: XSPI Capacitor set to 1/2
   11: XSPI Capacitor set to full capacitance
   see the product datasheet for more details

Bit 9 **VBRS**: VBAT charging resistor selection
   0: Charge VBAT through a 5 kΩ resistor.
   1: Charge VBAT through a 1.5 kΩ resistor.

Bit 8 **VBE**: VBAT charging enable
   0: VBAT battery charging disabled.
   1: VBAT battery charging enabled.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SDHILEVEL**: SMPS step-down converter voltage output for external supply.
   This bit is used when SMPSEXTHP is enabled. In this case SDHILEVEL has to be set to 1 to confirm the regulator settings.
   0: Reset value
   1: 1.8V

Bit 3 **SMPSEXTHP**: SMPS external power delivery selection
   0: SMPS normal operating mode, no power delivery to external circuits
   1: SMPS external operating mode, power delivery to external circuits

**Note**: Illegal combinations of SDHILEVEL, SMPSEXTHP, SDEN, LDOEN and BYPASS are described in Table 44.
Bit 2  **SDEN**: SMPS step-down converter enable  
   0: SMPS step-down converter disabled  
   1: SMPS step-down converter enabled. (Default)  

   **Note:**  Illegal combinations of SDHILEVEL, SMPSEXTHP, SDEN, LDOEN and  
   BYPASS are described in Table 44.

Bit 1  **LDOEN**: Low drop-out regulator enable  
   0: Low drop-out regulator disabled.  
   1: Low drop-out regulator enabled (default)  

   **Note:**  Illegal combinations of SDHILEVEL, SMPSEXTHP, SDEN, LDOEN and  
   BYPASS are described in Table 44.

Bit 0  **BYPASS**: Power management unit bypass  
   0: Power management unit normal operation.  
   1: Power management unit bypassed, voltage monitoring still active.  

   **Note:**  Illegal combinations of SDHILEVEL, SMPSEXTHP, SDEN, LDOEN and  
   BYPASS are described in Table 44.
6.8.5  PWR CPU control register 3 (PWR_CSR3)

This register allows controlling the CPU power.

Address offset: 0x010
Reset value: 0x0000 0000
After Standby Reset value: 0x0000 0200

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

Bit 9  **SBF**: System Standby flag
This bit is set by hardware and cleared only by a POR (Power-on Reset) or by setting the CPU CSSF bit.
0: System has not been in Standby mode
1: System has been in Standby mode

Bit 8  **STOPF**: STOP flag
This bit is set by hardware and cleared only by any reset or by setting the CPU CSSF bit.
0: System has not been in Stop mode
1: System has been in Stop mode

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

Bit 1  **CSSF**: Clear Standby and Stop flags (always read as 0)
This bit is cleared to 0 by hardware.
0: No effect.
1: flags (STOPF, SBF) are cleared.

Bit 0  **PDDS**: Power Down Deepsleep.
This bit allows CPU to define the Deepsleep mode
0: Stop mode when device enters Deepsleep.
1: Standby mode when device enters Deepsleep.
6.8.6 PWR control status register 4 (PWR_CSR4)

Address offset: 0x014
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

**Bit 1 VOSRDY**: VOS Ready bit for VCORE voltage scaling output selection.
When an internal regulator is used, this bit indicates that all the features allowed by the selected VOS can be used.*

*Note*: When Bypass mode is selected in the PWR control register (PWR_CSR2), VOSRDY bit is set to 1 by hardware whatever the VCORE level.

- 0: Not ready, voltage level below VOS selected level.
- 1: Ready, voltage level at or above VOS selected level.

**Bit 0 VOS**: Voltage scaling selection according to performance
These bits control the VCORE voltage level and allow to obtains the best trade-off between power consumption and performance:

- When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
- When decreasing performance, the system frequency must first be decreased before changing the voltage scaling.

- 0: VOS low level (default)
- 1: VOS high level

*Note*: Refer to Section "Electrical characteristics" of the product datasheet for more details.
6.8.7 PWR wakeup clear register (PWR_WKUPCR)

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 3 WKUPC4: Clear Wakeup pin flag for WKUP4
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF4 Wakeup pin flag (bit is cleared to 0 by hardware)

Bit 2 WKUPC3: Clear Wakeup pin flag for WKUP3
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF3 Wakeup pin flag (bit is cleared to 0 by hardware)

Bit 1 WKUPC2: Clear Wakeup pin flag for WKUP2
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF2 Wakeup pin flag (bit is cleared to 0 by hardware)

Bit 0 WKUPC1: Clear Wakeup pin flag for WKUP1
These bits are always read as 0.
0: No effect
1: Writing 1 clears the WKUPF1 Wakeup pin flag (bit is cleared to 0 by hardware)
### 6.8.8 PWR wakeup flag register (PWR_WKUPFR)

Address offset: 0x024  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

**Bit 3 WKUPF4**: Wakeup pin WKUP4 flag.  
This bit is set by hardware and cleared only by a Reset pin or by setting the WKUPC4 bit in the PWR wakeup clear register (PWR_WKUPCR).  
0: No wakeup event occurred  
1: A wakeup event was received from WKUP4 pin

**Bit 2 WKUPF3**: Wakeup pin WKUP3 flag.  
This bit is set by hardware and cleared only by a Reset pin or by setting the WKUPC3 bit in the PWR wakeup clear register (PWR_WKUPCR).  
0: No wakeup event occurred  
1: A wakeup event was received from WKUP3 pin

**Bit 1 WKUPF2**: Wakeup pin WKUP2 flag.  
This bit is set by hardware and cleared only by a Reset pin or by setting the WKUPC2 bit in the PWR wakeup clear register (PWR_WKUPCR).  
0: No wakeup event occurred  
1: A wakeup event was received from WKUP2 pin

**Bit 0 WKUPF1**: Wakeup pin WKUP1 flag.  
This bit is set by hardware and cleared only by a Reset pin or by setting the WKUPC1 bit in the PWR wakeup clear register (PWR_WKUPCR).  
0: No wakeup event occurred  
1: A wakeup event was received from WKUP1 pin
### 6.8.9 PWR wakeup enable and polarity register (PWR_WKUPEPR)

Address offset: 0x028  
Reset value: 0x0000 0000

(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WKUPPUPDn[1:0]**: Wakeup pin pull configuration for WKUPn, (n = 4 to 1)  
These bits define the I/O pad pull configuration used when WKUPENn = 1. The associated GPIO port pull configuration must be set to the same value or to 00.  
The Wakeup pin pull configuration is kept in Standby mode.  
00: No pull-up  
01: Pull-up  
10: Pull-down  
11: Reserved

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

Bits 11:8 **WKUPP[4:1]**: Wakeup pin polarity bit for WKUPn, (n = 4, 3, 2, 1)  
These bits define the polarity used for event detection on WKUPn external wakeup pin.  
0: Detection on high level (rising edge)  
1: Detection on low level (falling edge)

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WKUPEN[4:1]**: Enable Wakeup Pin WKUPn, (n = 4, 3, 2, 1)  
Each bit is set and cleared by software.  
0: An event on WKUPn pin does not wakeup the system from Standby mode.  
1: A rising or falling edge on WKUPn pin wakes-up the system from Standby mode.  
*Note: An additional wakeup event is detected if WKUPn+1 pin is enabled (by setting the WKUPENn bit) when WKUPn pin level is already high when WKUPn+1 selects rising edge, or low when WKUPn selects falling edge.*
6.8.10  PWR USB Type-C and Power Delivery register (PWR_UCPDR)

Address offset: 0x02C
Reset value: 0x0000 0002

(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

**Bit 1 UCPD_STBY**: UCPD Standby mode
When set, this bit is used to memorize the UCPD configuration in Standby mode.
This bit must be written to 1 just before entering Standby mode when using UCPD.
It must be written to 0 after exiting the Standby mode and before writing any UCPD registers.

**Bit 0 UCPD_DBDIS**: UCPD dead battery disable
After exiting reset, the USB Type-C “dead battery” behavior is enabled, which may have a pull-down effect on CC1 and CC2 pins. It is recommended to disable it in all cases, either to stop this pull-down or to handover control to the UCPD (the UCPD must be initialized before doing the disable).
0: UCPD dead battery pull-down behavior enabled on UCPDx_CC1 and UCPDx_CC2 pins
1: UCPD dead battery pull-down behavior disabled on UCPDx_CC1 and UCPDx_CC2 pins
6.8.11 PWR apply pull configuration register (PWR_APCR)

Address offset: 0x030
Reset value: 0x0003 0000

(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>I3CPB9_PU: Port PB9 I3C pull-up bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 30</td>
<td>I3CPB8_PU: Port PB8 I3C pull-up bit</td>
</tr>
<tr>
<td>Bit 29</td>
<td>I3CPB7_PU: Port PB7 I3C pull-up bit</td>
</tr>
<tr>
<td>Bit 28</td>
<td>I3CPB6_PU: Port PB6 I3C pull-up bit</td>
</tr>
<tr>
<td>Bit 27:18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 17</td>
<td>PO5_PUPD: Port O bit 5 pull-up/down configuration</td>
</tr>
<tr>
<td>Bit 16</td>
<td>PN7_PUPD: Port N bit 7 pull-up/down configuration</td>
</tr>
</tbody>
</table>

Bits 27:18 Reserved, must be kept at reset value.

Bit 17 PO5_PUPD: Port O bit 5 pull-up/down configuration
When this bit is set, a weak pull-up or pull down resistor is applied on PO5 following inverse logic applied on PO4.
If the PUO4 bit in PWR_PUCRO register is set and APC bit is set the week pull-down is applied on PO5.
If the PDO4 bit in PWR_PDCRO register is set and APC bit is set the week pull-up is applied on PO5.

Bit 16 PN7_PUPD: Port N bit 7 pull-up/down configuration
When this bit is set, a weak pull-up or pull-down resistor is applied on PN7 following inverse logic applied on PN6.
If the PUN6 bit in PWR_PUCRN register is set and APC bit is set the week pull-down is applied on PN7.
If the PDN6 bit in PWR_PDCRN register is set and APC bit is set the week pull-up is applied on PN7.

Bits 15:1 Reserved, must be kept at reset value.
Bit 0  **APC**: Apply pull-up and pull-down configuration

When this bit is set, the I/O pull-up and pull-down configurations defined in PO5_PUPD, PN7_PUPD bits and PUCRx, PDCRx registers are applied in Standby mode even after wakeup until APC bit is reset to 0.

When this bit is cleared, the I/O pull-up or pull-down configurations defined in PO5_PUPD, PN7_PUPD bits and PUCRx and PDCRx registers are not applied in Standby mode and IO becomes Hi-Z.
### 6.8.12 PWR port N pull-up control register (PWR_PUCRN)

Address offset: 0x034

Reset value: 0x0000 0000 (POR and System Reset)

(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:13 Reserved, must be kept at reset value.

**Bit 12** PUN12: Port N pull-up bit 12

When set, each bit activates the pull-up on PN12 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PD12 bit is also set.

Bits 11:7 Reserved, must be kept at reset value.

**Bit 6** PUN6: Port N pull-up bit 6

When set activates the pull-up on PN6 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PDN6 bit is also set.

Bits 5:2 Reserved, must be kept at reset value.

**Bit 1** PUN1: Port N pull-up bit 1

When set, each bit activates the pull-up on PN1 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding PD1 bit is also set.

**Bit 0** Reserved, must be kept at reset value.
### 6.8.13 PWR port N pull-down control register (PWR_PDCRN)

Address offset: 0x038  
Reset value: 0x0000 0000  
(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:13 Reserved, must be kept at reset value.

- **Bit 12** **PDN12**: Port N pull-down bit 12  
  When set activates the pull-down on PN12 when the APC bit is set in PWR_APCR.

Bits 11:9 Reserved, must be kept at reset value.

- **Bit 8** **PDN8N11**: Port N - PN8 to PN11 pull-down activation  
  When set, four pull-down resistors are activated on PN8 to PN11 when the APC bit is set in PWR_APCR.

- **Bit 7** Reserved, must be kept at reset value.

- **Bit 6** **PDN6**: Port N pull-down bit 6  
  When set activates the pull-down on PN6 when the APC bit is set in PWR_APCR.

Bits 5:3 Reserved, must be kept at reset value.

- **Bit 2** **PDN2N5**: Port N PN2 to PN5 pull-down activation  
  When set, four pull-down resistors are activated on PN2 to PN5 when the APC bit is set in PWR_APCR.

- **Bit 1** **PDN1**: Port N pull-down bit 1  
  When set activates the pull-down on PN1 when the APC bit is set in PWR_APCR.

- **Bit 0** **PDN0**: Port N pull-down bit 0  
  When set activates the pull-down on PN0 when the APC bit is set in PWR_APCR.
### 6.8.14 PWR port O pull-up control register (PWR_PUCRO)

Address offset: 0x03C  
Reset value: 0x0000 0000  
(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>PUO4</th>
<th>Res</th>
<th>Res</th>
<th>PUO1</th>
<th>PUO0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 4 PUO4**: Port O pull-up bit 4  
When set activates the pull-up on PO4 when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding bits PDO4 in PWR_PDCRO is also set.

Bits 3:2 Reserved, must be kept at reset value.

**Bits 1:0 PUOn**: (n = 1 to 0) Port O pull-up bits  
When set, each bit activates the pull-up on POy when the APC bit is set in PWR_APCR. The pull-up is not activated if the corresponding bits in PWR_PDCRO is also set.
### 6.8.15 PWR port O pull-down control register (PWR_PDCRO)

Address offset: 0x040  
Reset value: 0x0000 0000  
(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 4:0 PDOy: Port O pull-down bit y (y = 4 to 0)  
When set, each bit activates the pull-down on POy when the APC bit is set in PWR_APCR.
### 6.8.16 PWR port P pull-down control register (PWR_PDCRP)

Address offset: 0x044  
Reset value: 0x0000 0000  
(Reset value after Standby: the value of the register is kept in Standby mode.)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bits 31:13**: Reserved, must be kept at reset value.
  - **Bit 12 PDP12P15**: Port P12-P15 pull-down activation  
    When set, four pull-down resistors are activated on P8 to P11 when the APC bit is set in PWR_APCR.

- **Bits 11:9**: Reserved, must be kept at reset value.
  - **Bit 8 PDP8P11**: Port P8-P11 pull-down activation  
    When set, four pull-down resistors are activated on P8 to P11 when the APC bit is set in PWR_APCR.

- **Bits 7:5**: Reserved, must be kept at reset value.
  - **Bit 4 PDP4P7**: Port P4-P7 pull-down activation  
    When set, four pull-down resistors are activated on P4 to P7 when the APC bit is set in PWR_APCR.

- **Bits 3:1**: Reserved, must be kept at reset value.
  - **Bit 0 PDP0P3**: Port P0-P3 pull-down activation  
    When set, four pull-down resistors are activated on P0 to P3 when the APC bit is set in PWR_APCR.
## 6.8.17 PWR register map

### Table 53. Power control register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>0x000</th>
<th>0x004</th>
<th>0x008</th>
<th>0x00C</th>
<th>0x010</th>
<th>0x014</th>
<th>0x020</th>
<th>0x024</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>PWR_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x004</td>
<td>PWR_SR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x008</td>
<td>PWR_CSR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x00C</td>
<td>PWR_CSR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x010</td>
<td>PWR_CSR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x014</td>
<td>PWR_CSR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x020</td>
<td>PWR_WKUPCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x024</td>
<td>PWR_WKUPFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x028</td>
<td>PWR_WKUPEPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x030</td>
<td>PWR_APCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x034</td>
<td>PWR_PUCRN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>0x038</td>
<td>PWR_PDCRN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>
Table 53. Power control register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03C</td>
<td>PWR_PUCRO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>PWR_PDCRO</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>PWR_PDCRP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
7 Reset and clock control (RCC)

The RCC block manages the clock and reset generation for the whole microcontroller. The operating modes to which this section refers are defined in Section 6.6.1: Operating modes of the PWR block.

7.1 RCC main features

AHB-Lite bus interface

Reset block:
- Generation of local and system reset
- Bidirectional pin reset allowing to reset the microcontroller or external devices
- WWDG and IWDG reset supported
- Power-on (POR) and Brownout (BOR) resets initiated by the power control (PWR)

Clock generation block:
- Generation and dispatching of clocks for the complete device
- 3 separate PLLs using integer or fractional ratios
- Possibility of changing the PLL fractional ratios on-the-fly
- Smart clock gating to reduce power dissipation
- System clock protection
- 2 external oscillators:
  - High-speed external oscillator (HSE) supporting a wide range of crystals from 4 to 50 MHz frequency \(^{(a)}\)
  - Low-speed external oscillator (LSE) for the 32 kHz crystals
- 4 internal oscillators
  - High-speed internal oscillator (HSI)
  - 48 MHz RC oscillator (HSI48)
  - Low-power internal oscillator (CSI)
  - Low-speed internal oscillator (LSI)
- Buffered clock outputs for external devices
- Generation of two types of interrupts lines:
  - Dedicated interrupt lines for clock security management
  - One general interrupt line for other events
- Clock generation handling in Stop and Standby mode

\(^{(a)}\) Note that when the USBHSPHY is used, the HSE frequency must be 16, 19.2, 20, 24, 26 or 32 MHz.
7.2 **RCC block diagram**

*Figure 37 shows the RCC block diagram.*

*Figure 37. RCC block diagram*
7.3 RCC pins and internal signals

*Table 54* lists the RCC inputs and output signals connected to package pins or balls.

**Table 54. RCC input/output signals connected to package pins or balls**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NRST</td>
<td>I/O</td>
<td>System reset, can be used to provide reset to external devices</td>
</tr>
<tr>
<td>OSC32_IN</td>
<td>I</td>
<td>32 kHz oscillator input</td>
</tr>
<tr>
<td>OSC32_OUT</td>
<td>O</td>
<td>32 kHz oscillator output</td>
</tr>
<tr>
<td>OSC_IN</td>
<td>I</td>
<td>System oscillator input</td>
</tr>
<tr>
<td>OSC_OUT</td>
<td>O</td>
<td>System oscillator output</td>
</tr>
<tr>
<td>MCO1</td>
<td>O</td>
<td>Output clock 1 for external devices</td>
</tr>
<tr>
<td>MCO2</td>
<td>O</td>
<td>Output clock 2 for external devices</td>
</tr>
<tr>
<td>I2S_CKIN</td>
<td>I</td>
<td>External kernel clock input for digital audio interfaces: SPI/I2S, SAI, and ADF</td>
</tr>
</tbody>
</table>

The RCC exchanges a lot of internal signals with all components of the product, for that reason, *Table 55* only shows the most significant internal signals.

**Table 55. RCC internal input/output signals**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rcc_it</td>
<td>O</td>
<td>General interrupt request line</td>
</tr>
<tr>
<td>rcc_hsecss_it</td>
<td>O</td>
<td>HSE clock security failure interrupt</td>
</tr>
<tr>
<td>rcc_lsecss_it</td>
<td>O</td>
<td>LSE clock security failure interrupt</td>
</tr>
<tr>
<td>rcc_hsecss_fail</td>
<td>O</td>
<td>Event indicating that a HSE clock security failure is detected. This signal is connected to TIMERS.</td>
</tr>
<tr>
<td>rcc_lsecss_fail</td>
<td>O</td>
<td>Event indicating that a LSE clock security failure is detected. This signal is connected to TAMP, EXTI and PWR</td>
</tr>
<tr>
<td>nreset</td>
<td>I/O</td>
<td>Application reset</td>
</tr>
<tr>
<td>sys_rst</td>
<td>I/O</td>
<td>System reset</td>
</tr>
<tr>
<td>iwdg_out_rst</td>
<td>I</td>
<td>Reset line driven by the IWDG, indicating that a timeout occurred</td>
</tr>
<tr>
<td>wwdg_out_rst</td>
<td>I</td>
<td>Reset line driven by the WWDG, indicating that a timeout occurred</td>
</tr>
<tr>
<td>pwr_bor_rst</td>
<td>I</td>
<td>Brownout reset generated by the PWR block</td>
</tr>
<tr>
<td>pwr_por_rst</td>
<td>I</td>
<td>Power-on reset generated by the PWR block</td>
</tr>
<tr>
<td>pwr_vsw_rst</td>
<td>I</td>
<td>Power-on reset of the VSW domain generated by the PWR block</td>
</tr>
<tr>
<td>rcc_perx_rst</td>
<td>O</td>
<td>Reset generated by the RCC for the peripherals</td>
</tr>
<tr>
<td>pwr_wkup</td>
<td>I</td>
<td>Wakeup domain request generated by the PWR and used to restore the domain clocks</td>
</tr>
<tr>
<td>rcc_pwd_req</td>
<td>O</td>
<td>Low-power request generated by the RCC and used to set the circuit into low-power mode</td>
</tr>
</tbody>
</table>
### Table 55. RCC internal input/output signals (Continued)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cpu_sleep</td>
<td>I</td>
<td>Signals generated by the CPU, indicating if the CPU is in CRun, Sleep or CStop</td>
</tr>
<tr>
<td>cpu_deepsleep</td>
<td>I</td>
<td></td>
</tr>
<tr>
<td>perx_ker_ckreq</td>
<td>I</td>
<td>Signal generated by some peripherals in order to request the activation of their kernel clock</td>
</tr>
<tr>
<td>rcc_perx_ker_ck</td>
<td>O</td>
<td>Kernel clock signals generated by the RCC, for some peripherals</td>
</tr>
<tr>
<td>rcc_perx_bus_ck</td>
<td>O</td>
<td>Bus interface clock signals generated by the RCC for peripherals</td>
</tr>
<tr>
<td>rcc_bus_ck</td>
<td>O</td>
<td>Clocks for APB, AHB and AXI bridges generated by the RCC</td>
</tr>
<tr>
<td>rcc_cpu_ck</td>
<td>O</td>
<td>Clocks for the CPU, generated by the RCC</td>
</tr>
<tr>
<td>rcc_fclk</td>
<td>O</td>
<td></td>
</tr>
</tbody>
</table>
7.4 RCC reset block functional description

The RCC handles the reset generation for the complete product, using events coming from different sources:
- assertion of the NRST pin from an external device
- a failure on the supply voltage applied to VDD, VBAT
- an exit from Standby
- a watchdog timeout
- a software command

The reset scope depends on the source that generates the reset.

7.4.1 Reset from PWR block

The PWR block provides several reset signals to the RCC:
- The power-on/off reset signal (**pwr_por_rst**), which is asserted when the VDD supply is lower than the VPOR threshold.
- The brown-out reset signal (**pwr_bor_rst**), which is asserted when the VDD supply is lower than the VBOR threshold.
- The core reset signal (**pwr_vcore_ok**), which is asserted when the VDDCORE supply is not valid or available. Note that the VDDCORE voltage is switched off, when the product is in Standby. So when the system exits from Standby mode, the **pwr_vcore_ok** signal is asserted as long the VDDCORE voltage provided by the internal regulator is not valid. Note that when VDD supply is not valid, **pwr_vcore_ok** is asserted as well.
- The VSW domain reset signal (**pwr_vsw_rst**), which is asserted when the VSW supply is lower than the expected threshold.

*Note:* The **rcc_vcore_rst** is generated from the **pwr_vcore_ok** signal, the main difference between these two signals is that **rcc_vcore_rst** remains asserted until the option byte loading operation is completed.

Refer to *Table 56: Reset coverage summary* for details.
Figure 38. Simplified reset circuit
7.4.2 The system and application resets (sys_rst and nreset)

A system reset (sys_rst) resets most of the registers to their default values unless otherwise specified in the register description.

A system reset can be generated from one of the following sources:

- an assertion of the NRST pin (external reset)
- a reset from the power-on/off reset block (pwr_por_rst)
- a reset from the brownout reset block (pwr_bor_rst)

Refer to Section 6.5.2: Brownout reset (BOR) for a detailed description of the BOR function.

- a reset from the independent watchdogs (iwdg_out_rst)
- an exit from Standby (rcc_vcore_rst).
- a reset from the window watchdogs depending on WWDG configuration (wwdg_out_rst)
- a software reset from the Cortex®-M7 core
  It is generated via the SYSRESETREQ signal issued by the Cortex®-M7 core. This signal is also named SFTRESET in this document.
- A reset from the low-power mode security reset, depending on option byte configuration (lpwr_rst).
- An option byte reload request from the flash interface (obl_rst)

The application reset (nreset) is similar to the system reset, but is not asserted when the system exits from Standby.

Note: The SYSRESETREQ bit in Cortex®-M7 through the FPU application interrupt and reset control register, must be set to force a software reset on the device. Refer to the Cortex®-M7 with FPU technical reference manual for more details (see http://infocenter.arm.com).

7.4.3 The NRST reset

The NRST is active low. A pulse stretcher guarantees a minimum reset pulse duration of 20 μs (Refer to datasheet for details). In addition it is possible to extend the NRST assertion thanks to $C_R$ capacitor. It is not recommended to let the NRST pin unconnected.

When it is not used, connect this pin to ground via a 10 to 100 nF capacitor (CR in Figure 38). As shown in Figure 38, a filter is also present in order to suppress spurs coming from NRST PAD.

7.4.4 Low-power mode security reset (lpwr_rst)

To prevent critical applications from mistakenly enter a low-power mode, two low-power mode security resets are available. When enabled through nRST_STOP and nRST_STANDBY option bytes, a system reset is generated if the following conditions are met:

- The CPU accidentally enters Stop mode.
  This type of reset is enabled by resetting nRST_STOP user option byte. In this case, whenever the Stop mode entry sequence is successfully executed, a system reset is generated.
- The system accidentally enter Standby mode.
  This type of reset is enabled by resetting nRST_STANDBY user option byte. In this case,
case, whenever a Standby mode entry sequence is successfully executed, a system reset is generated. When the Standby mode is entered, a flag is also set in the power controller.

LPWRRSTF bit in the **RCC Reset status register (RCC_RSR)** indicates that a low-power mode security reset occurred (see line #7 in **Table 57**).

lpwr_rst is activated when a low-power mode security reset due to CPU occurred. Refer to **Section 5.4: FLASH option bytes** for additional information.

Refer to **Section 6: Power control (PWR)** for additional information and **Table 45: Operating mode summary** for the overview of the existing power modes.

### 7.4.5 Backup domain reset

A backup domain reset (pwr_vsw_rst) is generated when one of the following events occurs:

- Software reset, triggered by setting the VSWRST bit in the **RCC Backup domain control register (RCC_BDCR)**. All RTC registers and the RCC_BDCR register are set to their reset values, with the exception of LSEDRV, which is not changed if LSEON=1. The BKPSRAM is not affected.
- When the VSW voltage is outside the operating range. All RTC registers and the RCC_BDCR register are set to their reset values. In this case the content of the BKPSRAM is no longer valid.

Refer to **Section 6.4.4: Backup domain section of PWR block** and to section “System security” for additional information.

### 7.4.6 Coresight debug reset

The coresight debug components can be reset in different ways:

- Over the DAP by setting to 1 the bit CDBGRSTREQ of the control/status register of the debug port. This action asserts the debug reset request signal cdbgreq to the RCC. Then the RCC activates the debug reset (rcc_dbg_rst), a handshake signal cdbgreq is provided to the DAP request. The debug reset remains as long as the cdbgreq is to 1, refer to **Figure 38** for details.
- When VDDCORE power-on reset occurs (rcc_vcore_rst). This reset is activated after a power-on reset, or when the product exits from Standby.

Refer to **Section 1.7.1: CoreSight Debug Reset** for details.
7.4.7 Option bytes loading

As shown in Figure 38, the option bytes loading (OBL) sequence is triggered under the following conditions:

- After a power-on, or
- When the system exits from Standby
- When the flash interface requested a reload of option bytes

The system reset (sys_rst) can be released only when the option byte loading (OBL) is completed.

7.4.8 Peripheral resets

The application can individually reset any peripheral whenever requested. This can be done via the RCC_xxxxRSTR registers where “xxxx” is the name of the bus to which the peripheral is connected.

In order to reset a peripheral, the corresponding reset bit must be set to 1, and then set back to 0. There is no need to enable the peripheral clock in order to reset a peripheral.

Note also that the CRYP, SAES and HASH blocks can be reset in the case of a tamper event.

7.4.9 Reset coverage summary

Table 56: Reset coverage summary gives a detailed view of the reset coverage of the most important reset sources.

Note: When $V_{DD}$ is not valid, $V_{DDCORE}$ is also not valid.
### Table 56. Reset coverage summary\(^{(1)}\)

<table>
<thead>
<tr>
<th>Functions that are reset</th>
<th>Main reset lines</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\text{pwr_por_rst}) (2)</td>
</tr>
<tr>
<td>VDD domain</td>
<td>x</td>
</tr>
<tr>
<td>MCU</td>
<td>x</td>
</tr>
<tr>
<td>WWDG</td>
<td>x</td>
</tr>
<tr>
<td>IWDG</td>
<td>x</td>
</tr>
<tr>
<td>AXI/AHB Interconnections.</td>
<td></td>
</tr>
<tr>
<td>Debug components, including DBGMCU</td>
<td></td>
</tr>
<tr>
<td>Reset all the debug parts except the SWJ-DP function.</td>
<td></td>
</tr>
<tr>
<td>The SWJ-DP is reset by the NJTRST or (\text{rcc_vcore_rst}) resets.</td>
<td></td>
</tr>
<tr>
<td>The bit DCRT on DBGMCU_CR is reset with (\text{pwr_por_rst}).</td>
<td></td>
</tr>
<tr>
<td>Hardware system init.</td>
<td></td>
</tr>
<tr>
<td>It includes the reload of the option bytes.</td>
<td></td>
</tr>
<tr>
<td>(\text{RCC})</td>
<td></td>
</tr>
<tr>
<td>(\text{RCC\ Reset\ status\ register\ (RCC_RSR)})</td>
<td></td>
</tr>
<tr>
<td>(\text{RCC\ clock\ control\ and\ status\ register\ (RCC_CSR)})</td>
<td></td>
</tr>
<tr>
<td>(\text{RCC\ Backup\ domain\ control\ register\ (RCC_BDCR)})</td>
<td></td>
</tr>
<tr>
<td>Other RCC registers</td>
<td></td>
</tr>
<tr>
<td>(\text{PWR})</td>
<td></td>
</tr>
<tr>
<td>(\text{PWR_CSR1})</td>
<td></td>
</tr>
<tr>
<td>(\text{PWR_CSR2})</td>
<td></td>
</tr>
<tr>
<td>(\text{PWR_CSR3})</td>
<td></td>
</tr>
<tr>
<td>Individual bits of this register do not have the same reset condition, refer to the PWR section for details.</td>
<td></td>
</tr>
<tr>
<td>(\text{PWR_WKUPCR, PWR_WKUPFR, PWR_WKUPEPR})</td>
<td></td>
</tr>
<tr>
<td>Other registers</td>
<td></td>
</tr>
<tr>
<td>(\text{RTC})</td>
<td></td>
</tr>
<tr>
<td>(\text{BKPSRAM})</td>
<td></td>
</tr>
<tr>
<td>After a reset of the VSW domain, the backup regulator of the BKPSRAM is disabled, this function is controlled via the PWR block, BREN bit.</td>
<td></td>
</tr>
<tr>
<td>If the (\text{rcc_vsw_rst}) reset is due to a too low VSW voltage, the content of BKPSRAM is lost.</td>
<td></td>
</tr>
<tr>
<td>(\text{SBS})</td>
<td></td>
</tr>
<tr>
<td>(\text{SBS_RSSCMDR})</td>
<td></td>
</tr>
<tr>
<td>Other Peripherals</td>
<td></td>
</tr>
</tbody>
</table>

1. The 'x' means that the function is reset by the corresponding reset line, the '-' means that the function is not reset by the corresponding reset line.
2. The \(\text{pwr\_por\_rst}\) is asserted when the voltage applied to VDD is not valid. When \(\text{pwr\_por\_rst}\) is asserted, the \(\text{rcc\_vcore\_rst}\), \(\text{nrst}\), \(\text{sys\_rst}\) and the \(\text{reset}\), are asserted as well.
3. The rcc_vcore_rst is asserted when the voltage applied to VDD is not valid, or when the system exits from Standby (because VDDCORE was switched off). When rcc_vcore_rst is asserted, the sys_rst, and the pwr_dbg_rst are asserted as well.

4. When nreset is asserted, the sys_rst is also asserted.

### 7.4.10 Reset source identification

The CPU can identify the reset source by checking the reset flags in the RCC_RSR register. The CPU can reset the flags by setting RMVF bit.

*Table 57* shows how the status bits of the RCC_RSR register behave according to the situation that generated the reset. For example, when an IWDG timeout occurs (line #6), if the CPU is reading the RCC_RSR register during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

<table>
<thead>
<tr>
<th>#</th>
<th>Situations generating a reset</th>
<th>SBF (2)</th>
<th>LPWRRSTF</th>
<th>WWDRSTF</th>
<th>IWDRSTF</th>
<th>SFRSTF</th>
<th>PORRSTF</th>
<th>PINRSTF</th>
<th>BORRSTF</th>
<th>OBLRSTF</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Power-on reset (<em>pwr_por_rst</em>)</td>
<td>0 0 0 0</td>
<td>0 0 1 1 1</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Pin reset (NRST)</td>
<td>0 0 0 0</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Brownout reset (<em>pwr_bor_rst</em>)</td>
<td>0 0 0 0</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>System reset generated by CPU (SFTRESET)</td>
<td>0 0 0 0</td>
<td>1 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>WWDG reset (<em>wwdg_out_rst</em>)</td>
<td>0 0 1 0</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>IWDG reset (<em>iwdg_out_rst</em>)</td>
<td>0 0 0 1</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>CPU erroneously enters Stop or Standby mode</td>
<td>0 1 0 0</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>The flash interface requested a reload of option bytes</td>
<td>0 0 0 0</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>The product exits from Standby</td>
<td>1 0 0 0</td>
<td>0 0 1 0 0</td>
<td>1 0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Grayed cells highlight the register bits that are set.
2. The SBF bit is located into the PWR_CSR3 register of the PWR block

### 7.4.11 Power-on and wakeup sequences

For detailed diagrams refer to *Section 6.4.1: System supply startup* in the PWR section.

The time interval between the event that exits the product from a low-power and the moment where the CPU is able to execute code, depends on the system state and on its configuration. *Figure 39* shows the most usual examples.

**Power-on wakeup sequence**

The power-on wakeup sequence shown in *Figure 39* gives the most significant phases of the power-on sequence. It is the longest sequence since the circuit was not powered. Note that this sequence remains unchanged, whatever V_{BAT} was present or not.
Boot from pin reset

When a pin reset occurs, \( V_{DD} \) is still present. As a result:
- The regulator settling time is faster since the reference voltage is already stable.
- The HSI restart delay may be needed if the HSI was not enabled when the NRST occurred, otherwise this restart delay phase is skipped.
- The Flash memory power recovery delay can also be skipped if the Flash memory was enabled when the NRST occurred.

**Note:** The boot sequence is similar for `pwr_bor_rst`, `lpwr_rst`, `STFRESET`, `iwdg_out_rst` and `wwdg_out_rst`.

Boot from system Standby

When waking up from system Standby, the reference voltage is stable since \( V_{DD} \) has not been removed. As a result, the regulator settling time is fast. Since \( V_{CORE} \) was not present, the restart delay for the HSI, the Flash memory power recovery and the option byte reloading cannot be skipped.

Restart from system Stop

When restarting from Stop, \( V_{DD} \) is still present. As a result, the sequence is mainly composed of two steps:
1. Regulator settling time to reach VOS low (default voltage).
2. HSI/CSI restart delay. This step can be skipped if HSIKERON or CSIKERON bit is set to 1 in the **RCC register map**.

**Figure 39. Boot sequences versus system states**
7.5 RCC clock block functional description

The RCC provides a wide choice of clock generators:

- HSI (high-speed internal oscillator) clock: ~ 8, 16, 32 or 64 MHz
- HSE (high-speed external oscillator) clock: 4 to 50 MHz
- LSE (low-speed external oscillator) clock: 32 kHz
- LSI (low-speed internal oscillator) clock: ~ 32 kHz
- CSI (low-power internal oscillator) clock: ~4 MHz
- HSI48 (high-speed internal oscillator) clock: ~48 MHz

The RCC offers a high flexibility for the application to select the appropriate clock for CPU and peripherals, in particular for peripherals that require a specific clock such as SPI/I2S and SAI.

To optimize the power consumption, each clock source can be switched ON or OFF independently.

The RCC provides up to 3 PLLs; each of them can be configured in integer, with or without SSCG, or fractional mode.

As shown in the Figure 40, the RCC offers two clock outputs (MCO1 and MCO2), with a great flexibility on the clock selection and frequency adjustment.

The SCGU block (system clock generation unit) contains several prescalers used to configure the CPU and bus matrix clock frequencies.

The PKSU block (peripheral kernel clock selection unit) provides several dynamic switches allowing a large choice of kernel clock distribution to peripherals.

The PKEU (peripheral kernel clock enable unit) and SCEU (system clock enable unit) blocks perform the peripheral kernel clock gating, and the bus interface/cores/bus matrix clock gating, respectively.
Figure 40. Top-level clock tree

The selected input can be changed on-the-fly without spurs on the output signal.

* Represents the selected mux input after a system reset.
7.5.1 Clock naming convention

The RCC provides clocks to the complete circuit. To avoid misunderstanding, the following terms are used in this document:

- **Peripheral clocks**
  
The peripheral clocks are the clocks provided by the RCC to the peripherals. Two kinds of clock are available:
  - bus interface clocks
  - kernel clocks

A peripheral receives from the RCC a bus interface clock in order to access its registers, and thus control the peripheral operation. This clock is generally the AHB, APB or AXI clock depending on which bus the peripheral is connected to. Some peripherals only need a bus interface clock (such as RNG, TIMx).

Some peripherals also require a dedicated clock to handle the interface function. This clock is named kernel clock. As an example, peripherals such as SAI must generate specific and accurate master clock frequencies, which require dedicated kernel clock frequencies. Another advantage of decoupling the bus interface clock from the specific interface needs, is that the bus clock can be changed without reprogramming the peripheral.

- **CPU clock**
  
The CPU clock is the clock provided to the CPU. It is derived from the system clock (\(sys_{\text{ck}}\)).

- **Bus matrix clocks**
  
The bus matrix clocks are the clocks provided to the different bridges (APB, AHB or AXI). These clocks are derived from the system clock (\(sys_{\text{ck}}\)).

7.5.2 Oscillators description

The table hereafter shows the oscillator states versus system modes, when the oscillators are enabled via registers. The term “available” means that the resource can be used if activated via registers.

<table>
<thead>
<tr>
<th>System modes</th>
<th>VDD domain</th>
<th>VSW domain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HSE</td>
<td>HSI</td>
</tr>
<tr>
<td>Exit from system reset</td>
<td>OFF</td>
<td>ON</td>
</tr>
<tr>
<td>Exit from system Stop</td>
<td>OFF</td>
<td>ON (1)</td>
</tr>
<tr>
<td>In Run/Sleep</td>
<td>Available</td>
<td>Available (3)</td>
</tr>
<tr>
<td>In Stop</td>
<td>OFF</td>
<td>Available (4)</td>
</tr>
<tr>
<td>In Standby</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>In VBAT</td>
<td>OFF</td>
<td>OFF</td>
</tr>
</tbody>
</table>

1. If STOPWUCK = 0 or STOPKERWUCK = 0
2. If STOPWUCK = 1 or STOPKERWUCK = 1
3. HSI can remain activated in Stop if HSIKERON = 1, or if a peripheral selecting HSI, generates a kernel clock request.
4. CSI can remain activated in Stop if CSIKERON = 1, or if the peripheral selecting CSI, generates a kernel clock request.
HSE oscillator

The HSE block allows the application to provide a very accurate high-speed clock for the product. The HSE can generate an internal clock from two possible sources:
- external clock source
- external crystal/ceramic resonator

**Figure 41. HSE/LSE clock source**

External clock source (HSE bypass)

In this mode the oscillator is not used, and an external clock source must be provided to OSC_IN pin. The external clock can be low swing (analog) or digital. The OSC_OUT pin must be left HI-Z (see Figure 41).

The external clock signal can be digital or analog (square, sinus or triangle). An analog clock signal with a reduced amplitude, is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%. Refer to the datasheet for additional information.

This mode is selected when HSEBYP = 1, and HSEON = 1. In case of an analog clock input (low swing) the HSEEXT must be set to 0, for a digital clock input, HSEEXT bits must be set to 1.

**Figure 42. HSE clock generation**
External crystal/ceramic resonator

A crystal/resonator can be connected as shown in Figure 41: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is enabled by setting the HSEBYP bit to 0 and HSEON bit to 1.

HSE ready logic

The HSERDY flag indicates when a valid clock is available at HSE output (hse_ck). When the HSE is enabled (HSEON set to '1'), the HSERDY flag goes to '1' when 256 valid cycles of HSE have been detected. The hse_ck clock is not released until HSERDY goes to 1.

An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

HSE controls

The HSE can be switched ON and OFF through the HSEON bit.

The hardware does not allow modifying HSEON bit if one of the condition is met:

- The HSE is used directly (via SW switch) as system clock.
- The HSE is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via SW switch).
- The bit XSPICKP is set to 1.
- The bit FMCKCP is set to 1.

The hardware does not allow changing the values of HSEBYP, and HSEEXT when HSEON = 1. Bits HSEBYP, HSEEXT and HSEON are located into the RCC register map.

The HSE is automatically disabled by hardware when the system enters Stop or Standby mode (See Table 58: Oscillator states versus system modes).

In addition, the HSE clock can be driven to the MCO1 and MCO2 outputs and used as clock source for other application components.
**Programming sequence**

In order to initialize the HSE, the application must follow the sequence hereafter:

- Make sure the HSE is not directly or indirectly used as system clock, if it is the case, use HSI or CSI as clock source for system clock.
- Make sure that XSPICKP and FMCCKP are set to 0
- Disable the HSE by writing to 0 the HSEON bit,
- Check that the HSE is disabled by waiting that the bit HSERDY is set to 0,
- If the oscillator mode is needed:
  - Select the oscillator mode by setting HSEBYP to 0,
- If an external clock is connected to OSC_IN:
  - Select the bypass mode by setting HSEBYP to 1,
  - If the input clock is a full-swing digital signal, set HSEEXT to 1
  - If the input clock is a low-swing signal, set HSEEXT to 0
- Enable again the HSE by writing the HSEON bit to 1,
- Wait for HSERDY = 1, then the HSE is ready for use.

**LSE oscillator**

The LSE block allows the application to provide a very accurate low-frequency clock for the product. The LSE can generate an internal clock from two possible sources:

- external user clock
- external crystal/ceramic resonator

**External clock source (LSE bypass)**

In this mode, the oscillator is not used, and an external clock source must be provided to OSC32_IN pin. The OSC32_OUT pin must be left Hi-Z (see [Figure 41](#)).

The external clock signal can have a frequency up to 1 MHz and be digital or analog (square, sinus or triangle). An analog clock signal with a reduced amplitude is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%. Refer to the datasheet for additional information.

This mode is selected by setting the LSEBYP and LSEON bits to 1. In case of an analog clock input (low swing) the LSEEXT must be set to 0, for a digital clock input, LSEEXT bits must be set to 1.

![Figure 43. LSE clock generation](#)
External crystal/ceramic resonator source (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It has the advantage to provide a low-power highly accurate clock source to the real-time clock (RTC) for clock/calendar or other timing functions. A crystal/resonator can be connected as shown in Figure 41: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is selected by setting LSEBYP bit to 0 and LSEON bit to 1.

The LSE also offers a programmable driving capability (LSEDRV[1:0]) that can be used to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to insure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

---

**Warning:** It is not allowed to change the driving capability when the LSE is enabled. The LSE behavior is not guaranteed in that case.

---

LSE ready logic

The LSE offers a LSERDY flag which indicates whether the LSE clock is available or not. When the LSE is enabled (LSEON set to 1) the LSERDY flag goes to 1 when certain amount of valid cycles of LSE clock have been detected. The lse_ck clock is not released until LSERDY goes to 1.

When LSEBYP = 0, the RCC waits 4096 clocks cycles before activating the LSERDY flag, when LSEBYP = 1, the RCC waits 8 clocks cycles.

An interrupt, can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

LSERDY bit is located into the RCC Backup domain control register (RCC_BDCR).

LSE controls

The LSE control bits LSEBYP, LSEEXT, LSEDRV[1:0] and LSEON are located into the RCC Backup domain control register (RCC_BDCR). This register is write-protected by DBP bit of PWR_CR1 register. In order to modify the RCC_BDCR register, the bit DBP must be set 1.

The LSE oscillator is switched ON and OFF using the LSEON bit.

The LSE remains enabled when the system enters Stop, Standby or VBAT mode (See Table 58: Oscillator states versus system modes).

The hardware does not allow changing the values of LSEBYP, and LSEEXT when LSEON = 1.

In addition, the LSE clock can be driven to the MCO1 output and used as clock source for external components.
LSE Programming sequence

In order initialize the LSE, the application must follow the sequence hereafter:

- Set DBP bit of PWR.CR1 to 1 in order to allow write accesses to RCC.BDCR register.
- Disable the LSE by writing to 0 the LSEON bit.
- Check that the LSE is disabled by waiting that the bit LSERDY is set to 0,
- if the oscillator mode is needed:
  - Select the oscillator mode by setting LSEBYP to 0,
  - Configure LSEDRV[1:0],
- if an external clock is connected to OSC32_IN:
  - Select the bypass mode by setting LSEBYP to 1,
  - If the input clock is a full-swing digital signal, set LSEEXT to 1
  - If the input clock is a low-swing signal, set LSEEXT to 0
- Enable again the LSE by writing the LSEON bit to 1,
- Wait for LSERDY = 1, then the LSE is ready for use.
- If RCC_BDCR register no longer need to be changed, set DBP bit of PWR.CR1 to 0 in order to write-protect accesses.

If the RTC is used, the LSE bypass must not be configured in digital mode but in low swing analog mode (default value after reset).
HSI oscillator

The HSI block provides the default clock to the product. The HSI is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32 or 64 MHz. This predivider is controlled by the HSIDIV.

The HSI advantages are the following:
- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

HSI controls

The HSI can be switched ON and OFF using the HSION bit.

The hardware does not allow modifying HSION bit if one of the condition is met:
- The HSI is used directly (via software mux) as system clock.
- The HSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).
- The bit XSPICKP is set to 1.
- The bit FMCCKP is set to 1.

Note that the HSIDIV cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case the hardware does not update the HSIDIV with the new value. However it is possible to change the HSIDIV if the HSI is used directly as system clock.

The HSIRDY flag indicates if the HSI is stable or not. At startup, the HSI output clock is not released until this bit is set by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (refer to Section: CSS on HSE).

The HSI can be disabled or not when the system enters Stop mode (See Table 58: Oscillator states versus system modes).

In addition, the HSI clock can be driven to the MCO1 output and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals, the application must take into account the following parameters:
- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency accuracy.

HSION, HSIRDY and HSIDIV bits are located in the RCC register map.

HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of ACC_\text{HSI} (refer to the product datasheet for more information).

After a power-on reset, or at the exit of Standby mode, the factory calibration value is loaded in the HSICAL[11:0] bits.
If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using the HSITRIM[6:0] bits.

*Note:* HSICAL[11:0] and HSITRIM[6:0] bits are located in the RCC HSI calibration register (RCC_HSICFGR).

**CSI oscillator**

The CSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

The CSI advantages are the following:

- low-cost clock source since no external crystal is required
- faster startup time than HSE (a few microseconds)
- very low-power consumption.

The CSI provides a clock frequency of about 4 MHz, while the HSI is able to provide a clock up to 64 MHz.

CSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

**CSI controls**

The CSI can be switched ON and OFF through the CSION bit. The CSIRDY flag indicates whether the CSI is stable or not. At startup, the CSI output clock is not released until this bit is set by hardware.

The hardware does not allow modifying CSION bit if one of the condition is met:

- The CSI is used directly (via software mux) as system clock.
- The CSI is selected as reference clock for PLL1, with PLL1 enabled and selected to provide the system clock (via software mux).
- The bit XSPICKP is set to 1.
- The bit FMCCKP is set to 1.

The CSI can be disabled or not when the system enters Stop mode (See Table 58: Oscillator states versus system modes).

In addition, the CSI clock can be driven to the MCO2 output and used as clock source for other application components.
Even if the CSI settling time is faster than the HSI, care must be taken when the CSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request and the moment where the clock is really available,
- the frequency precision.

**Note:** CSION and CSIRDY bits are located in the RCC register map.

### CSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations, this is why each device is factory calibrated by STMicroelectronics to achieve accuracy of ACCCSI (refer to the product datasheet for more information).

After a power-on reset, or at the exit of Standby mode, the factory calibration value is loaded in the CSICAL[7:0] bits.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the CSI frequency using the CSITRIM[5:0] bits.

**Note:** Bits CSICAL[7:0] and CSITRIM[5:0] are located into the RCC CSI calibration register (RCC_CSICFRG).

![Figure 45. CSI calibration flow](MSv48152V1)

### HSI48 oscillator

The HSI48 is an RC oscillator delivering a 48 MHz clock that can be used directly as kernel clock for some peripherals.

The HSI48 oscillator mainly aims at providing an accurate clock to the USB Full-Speed peripherals by means of a special clock recovery system (CRS) circuitry. The CRS can use the USB SOF signal, the LSE or an external signal to automatically adjust the oscillator frequency on-the-fly, with a very small granularity. It is possible to read the calibration value provided to the HSI48 via RCC clock recovery RC register (RCC_CRRCR). The HSI48TRIM value can be adjusted through the CRS block, for more details, refer to Section 9: Clock recovery system (CRS).
The HSI48 oscillator is disabled as soon as the system enters Stop or Standby mode. When the CRS is not used, this oscillator is free running and thus subject to manufacturing process variations. This is why each device is factory calibrated by STMicroelectronics to achieve an accuracy of $\text{ACC}_{\text{HSI48}}$ (refer to the product datasheet for more information).

The HSI48RDY flag indicates whether the HSI48 oscillator is stable or not. At startup, the HSI48 output clock is not released until this bit is set by hardware.

After a power-on reset, or at the exit of Standby mode, the factory calibration value is loaded into the HSI48.

The HSI48 can be switched ON and OFF using the HSI48ON bit.

The HSI48 clock can also be driven to the MCO1 multiplexer and used as clock source for other application components.

Note: HSI48ON and HSI48RDY bits are located in the RCC register map.

**LSI oscillator**

The LSI acts as a very-low-power clock source that can be kept running when the system is in Stop or Standby mode for the independent watchdog (IWDG) and auto-wakeup unit (AWU). The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched ON and OFF using the LSION bit. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced ON and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode (See *Table 58: Oscillator states versus system modes*).

At LSI startup, the clock is not provided until the hardware sets the LSIRDY bit. An interrupt can be generated if enabled in the RCC clock source interrupt enable register (RCC_CIER).

In addition, the LSI clock can be driven to the MCO2 output and used as a clock source for other application components.

Note: Bits LSION and LSIRDY bits are located into the RCC clock control and status register (RCC_CSR).
7.5.3 Clock security system (CSS)

The RCC offers a clock security system for the HSE and LSE oscillators. The clock security systems are capable of detecting a failure on HSE and LSE oscillators. The figure hereafter shows how this function interact with other blocks.

Figure 47. LSE and HSE CSS function

CSS on HSE

The clock security system can be enabled by software via the HSECSSON bit. The HSECSSON bit can be enabled even when the HSEON is set to 0.

The CSS on HSE is activated when the HSE is enabled and ready, and the software sets the HSECSSON bit to 1. The CSS on HSE is no longer working when the HSE is disabled. For example, this function does not work when the system is in Stop mode.

It is not possible to clear directly the HSECSSON bit by software. The HSECSSON bit is cleared by hardware when a system reset occurs or when the system enters Standby mode (see Section 7.4.2: The system and application resets (sys_rst and nreset)).
When the CSS on HSE is enabled, the following actions are done by the RCC if a failure is detected:

- The signal `rcc_hsecss_fail` is asserted.
- If the STOPWUCK = 0, the HSI clock is enabled and the SW switch automatically selects HSI as system clock.
- If the STOPWUCK = 1, the CSI clock is enabled and the SW switch automatically selects CSI as system clock.
- The HSE is then automatically disabled.
- If the HSE output was used as clock source for PLLs, the PLLs are also disabled.
- The XSPI[2:1]SEL switches go to recovery position.
- The FMCSEL switch goes to recovery position:
  - if PLL1 has been disabled and FMCSEL is in position 1.
  - if PLL2 has been disabled and FMCSEL is in position 2.
  See details on Section : Clock distribution for SDMMCs, FMC, and XSPIs.
- The clock failure event (`rcc_hsecss_fail`) is also sent to the break inputs of advanced-control timers (TIM1, TIM15, TIM16 and TIM17).
- A NMI interrupt is generated to inform the software about the failure (CSS interrupt: `rcc_hsecss_it`), thus allowing the MCU to perform rescue operations. The CSS interrupt is asserted until the HSECSSF bit is cleared. The HSECSSF flag can be cleared by setting the HSECSSC bit in the RCC clock source interrupt clear register (`RCC_CICR`).
- A tamper event can also be triggered, in order to clear content of the backup registers, and backup RAM.

CSS on LSE

A clock security system on the LSE oscillator can be enabled by software by programming the LSECSSON bit in the `RCC Backup domain control register (RCC_BDCR)`. This bit is disabled by hardware if one of the following condition is met:

- After a VSW hardware reset (`pwr_vsw_rst`),
- After a VSW software reset via VSWRST bit

The software can also disable the clock security system after a LSE failure detection. The CSS on LSE works in all modes (Run, Stop and Standby) including VBAT.

The LSECSS provides a re-arm feature, offering to the software the possibility to re-arm the LSECSS and re-enable the LSE clock when a failure has been detected. This feature allows the application to decide if the LSE is to be provided again to the RTC even if a failure occurred, or if another action must be performed. For example the application can decide to reset the VSW domain only if a certain amount of consecutive LSE failures occurred, within a time window.

The LSECSS offers two flag signals:

- The LSECSSD, capable of retaining an LSE failures even in VBAT mode.
- The LSECSSF used to generate interrupt in case of LSE failure. This flag is not affected by a failure detected when the product is in VBAT mode.
The sequence hereafter describes the LSE enabling sequence with the clock security system enabled.

- Follow the LSE enable procedure given in the section LSE Programming sequence, except the last step
- Select the LSE clock via RTCSEL[1:0]
- Set the LSECSSON bit to 1
- If RCC_BDCR register no longer needs to be changed, set DBP bit of PWR_CR1 to 0 in order to write-protect accesses.

Note: The LSECSSON bit must be enabled after the LSE is enabled (LSEON bit set by software) and ready (LSERDY set by hardware), and after the RTC clock has been selected through the RTCSEL bit.

If a failure is detected on the LSE, the hardware does the following:

- The LSE clock is no more delivered to the RTC.
- The values of RTCSEL, LSECSSON and LSEON bits are not changed by the hardware.
- A failure event is generated (rcc_lsecss_fail). This event allows the system to wake-up from Standby, but also to protect the backup registers and BKPSRAM via TAMP block. This event is also generated in VBAT mode.
- The interrupt flag LSECSSF is activated (except in VBAT mode) in order to generate an interrupt (rcc_lsecss_it).
- The LSECSSD flag is activated as well, retaining the first LSE failure even in VBAT.

In the software side, different actions can be taken according to the application requirements. Three different cases are described hereafter in order to illustrate the hardware behavior, they can also be combined. The application can also decide to handle LSE failure differently.

Case A:

The application no longer wants to use LSE when a failure is detected:

- Unlock RCC_BDCR register by setting the DBP bit of PWR_CR1 to 1
- Disable the CSS function (this step is mandatory)
  - Clear the LSECSSF bit, if interrupt was enabled for this event
  - Set the LSECSSON bit to 0
  - Set the LSEON bit to 0, in order to disable the LSE
- Change the clock source for the RTC if needed:
  - Set the RTCEN bit to 0 to disable the RTC clock
  - Enable the new clock source for the RTC
  - Select the proper clock source via RTCSEL
  - Set the RTCEN bit to 1 to enable the RTC clock
- The application must perform specific actions consecutive to TAMPER events if enabled
- Lock RCC_BDCR register by setting the DBP bit of PWR_CR1 to 0

Case B:
The application wants to re-initialize the VSW domain:
- Unlock RCC_BDCR register by setting the DBP bit of PWR_CR1 to 1
- Perform a VSW reset by setting VSWRST bit to 1, then back to 0 (it is in reality a pulse of one period of sys_ck)
- The application must perform specific actions consecutive to TAMPER events if enabled
- Re-initialize all components of the VSW domain.
- Lock RCC_BDCR register by setting the DBP bit of PWR_CR1 to 0

**Case C:**
The application tries to re-use LSE when a failure is detected:
- If the number of failures in a given time window is higher than a given threshold then go to case A or B. Otherwise continue to next step.
- Unlock RCC_BDCR register by setting the DBP bit of PWR_CR1 to 1
- Clear the LSECSSF bit if interrupt was enabled for this event.
- The application must perform specific actions consecutive to TAMPER events if enabled
- Set the LSECSSON bit to 0.
- Re-arm the LSECSS function by writing the LSECSSRA bit to 1, then back to 0.
- Wait for LSERDY set 1. The LSERDY flag must go to 1 after the oscillator settling time delay plus, 4096 periods of LSE clock. If this is not the case, it probably means that the LSE failure is permanent. It is not possible to set LSECSSON to 1, it is recommended to execute case A or B.
- Set the LSECSSON bit to 1.
- When LSECSSON is seen to 1, the LSE is enabled, and protected by LSECSS.
- Lock RCC_BDCR register by setting the DBP bit of PWR_CR1 to 0

All bits used in this sequences (except DBP) are located into **RCC Backup domain control register (RCC_BDCR)**.

### 7.5.4 Clock output generation (MCO1/MCO2)

Two microcontroller clock output pins MCO1 and MCO2, are available. A clock source can be selected for each output. The selected clock can be divided thanks to configurable prescaler (refer to **Figure 40** for additional information on signal selection).

MCO1 and MCO2 outputs are controlled via MCO1PRE[3:0], MCO1SEL[2:0], MCO2PRE[3:0] and MCO2SEL[2:0] located in the **RCC clock configuration register (RCC_CFGR)**.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

The dividers MCO1PRE and MCO2PRE, provide a clock with a duty cycle of 50% for even divisions values. More generally the duty cycle is given by the following formula:

\[
DC(\%) = \frac{\text{FLOOR}\left(\frac{\text{MCOxPRE}}{2}\right)}{\text{MCOxPRE}} \times 100
\]
For MCOxPRE = 2 to 15.

Note that the MCO1 and MCO2 outputs are available in Run and Stop modes.

**Caution:** The clock provided to the MCOs outputs must not exceed the maximum PAD speed, refer to the product datasheet for information about the supported PAD speed.

The table hereafter show the signals available on each MCO output.

<table>
<thead>
<tr>
<th>Pos.</th>
<th>Clock source</th>
<th>Pos.</th>
<th>Clock source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hsi_ck</td>
<td>0</td>
<td>sys_ck</td>
</tr>
<tr>
<td>1</td>
<td>lse_ck</td>
<td>1</td>
<td>pll2_p_ck</td>
</tr>
<tr>
<td>2</td>
<td>hse_ck</td>
<td>2</td>
<td>hse_ck</td>
</tr>
<tr>
<td>3</td>
<td>pll1_q_ck</td>
<td>3</td>
<td>pll1_p_ck</td>
</tr>
<tr>
<td>4</td>
<td>hsi48_ck</td>
<td>4</td>
<td>csi_ck</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>5</td>
<td>lsi_ck</td>
</tr>
</tbody>
</table>

Table 59. Oscillator states versus system modes
7.5.5 PLL description

The RCC features three PLLs:
- PLL1, generally used to provide clocks to the CPU, busses, and some peripherals
- PLL2, generally used to provide clocks to XSPI and SDMMC, and some peripherals
- PLL3, dedicated to the kernel clock generation for peripherals

The PLLs integrated into the RCC are completely independent. They offer the following features:
- A VCO supporting two modes:
  - a wide-range
  - a low-range used for instance in audio application cases
- Input frequency range:
  - 2 to 16 MHz for the VCO in wide-range mode
  - 1 to 2 MHz for the VCO in low-range mode
- VCO frequency range:
  - 384 to 1672 MHz for the VCO in high-range mode (VCOH), before the divider.
  - 150 to 420 MHz for the VCO in low-range mode (VCOL)
- Capability to work either in integer with or without SSGC, or in fractional mode
- 13-bit sigma-delta modulator, allowing to fine-tune the VCO frequency by steps of 10 to 0.3 ppm
- The sigma-delta modulator can be updated on-the-fly without generating frequency overshoots on PLLs outputs.
- PLLs offer up to 5 outputs with post-dividers.

For better flexibility, all PLLs offer two VCOs (VCOH and VCOL), the application can select the wanted VCO via PLLxVCOSEL bit.

The frequency of the reference clock provided to the PLLs (refx_ck) must range from 1 to 16 MHz. The DIVMx dividers of the RCC PLLs clock source selection register (RCC_PLLCKSELR) must be properly programmed in order to match this condition. In addition, the PLLxRGE[1:0] field of the RCC PLLs configuration register (RCC_PLLCFGR) must be set according to the reference input frequency to guarantee an optimal performance of the PLL.

The user application can then configure the proper VCO. VCOL must be chosen when the reference clock frequency is lower than 2 MHz.

To reduce the power consumption, it is recommended to configure the VCO output to the smaller range.
The PLLs can be enabled by setting PLLxON to 1. The PLLxRDY bits indicate that the PLL is ready (means locked).

The DIVN loop divider must be programmed to achieve the expected frequency at VCO output. The VCO output range must be respected.

PLLs have a divider-by-2, at VCOH output, insuring that the clock provided to the post-dividers has a duty-cycle of 50%.

The frequency of the clocks provided by the PLLs can also be adjusted thanks to post-dividers DIVP, DIVQ, DIVR, DIVS and DIVT. The post-divider values can be changed without disabling the PLLs, if their respective enable bits are set to 0.

The post-dividers provide clocks with 50% duty-cycle in the following conditions:
- When the post-dividers are dividing the clock by an even value.
- When VCOH is used and post-dividers are bypassed.
- When VCOH is selected and post-dividers DIVS and DIVT are used.

PLL programming recommendations

This section is providing a list of recommendations to follow in order to guarantee a good usage of the PLLs. Note that programming examples are given in Section 1.6.2: PLL programming.

- Before enabling the PLLs, the user must ensure that the reference frequency \( \text{refx}_\text{ck} \) provided to the PLL is stable and in the good range.
- When one or several PLL outputs are not used, the application must set the corresponding enable bit DIV[y]ENx and the divider value DIV[y]x to 0 (with x = 1, 2 or 3, and y = P, Q, R, S or T).
- In order to ensure the good behavior of the PLL, the bits PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN and PLLxTEN must be set to 0 before disabling the corresponding PLL. In the same way, the bits PLLxPEN, PLLxQEN, PLLxREN,
PLLxSEN and PLLxTEN can be set to 1 only if the corresponding bit PLLON is already set to 1 and the PLL is ready.

Refer to section *Section 7.7.1: PLL programming procedure* for additional information.

**PLL protections**

- The RCC hardware does not allow changing DIVMx, DIVN, PLLxRGE, PLLxVCOSEL when the corresponding PLLx is ON and it is also not possible to change PLLSRC when one of the PLL is ON.
- The RCC hardware does not allow changing of the post-dividers’ DIVy when the corresponding PLLxyEN bit is set to 1 (with x = 1, 2 or 3, and y = P, Q, R, S or T).
- The hardware prevents writing PLL1ON or PLL1PEN to 0 if the PLL1 is currently used to deliver the system clock. There are other hardware protections on the clock generators (refer to *Section 7.5.7: Clock protection*).

The PLLs are disabled by hardware when the system enters Stop or Standby mode.

PLLs using HSE as reference clock are also disabled by hardware if an HSE failure is detected.

*Note: The refx_ck clock is provided to the PLLx when the corresponding PLLON bit is set to 1.*

The PLLs can work in 3 different modes:

- In integer mode
- In fractional mode
- In spread spectrum mode

**Using the PLLs in integer mode**

The PLL is working in integer mode when the sigma-delta modulator (SDM) is loaded with a 0, and the bit PLLxSSCGEN = 0.

In order to load 0 into the SDM, the user has to perform the following sequence:

- Write PLLxFRACLE of *RCC PLLs configuration register (RCC_PLLCFGR)* to 0
- Write FRACN[12:0] to 0,
- Write PLLxFRACLE to 1
- Wait at least 5 µs

The bits FRACN[12:0] are located in the RCC_PLLxFRACR registers.

The VCO frequency ($F_{VCO}$) and output frequencies expressions are given by the following table:
Using the PLLs in fractional mode

The fractional mode is activated when the value loaded into the SDM is different from 0, and the bit PLLxSSCGEN = 0.

The SDM value can be updated at anytime by the application by executing the following sequence:
- Write PLLxFRACLE bit to ‘0’
- Write into FRACN[12:0] the new fractional value,
- Write PLLxFRACLE to ‘1’. The new FRACN value is loaded into the SDM when PLLxFRACLE bit goes from 0 to 1.
- Wait at least 5 µs

The sigma delta modulator is designed in order to minimize the jitter impact while allowing very small frequency steps.

When the PLL is used in fractional mode, the DIVN divider must be initialized before enabling the PLL.
This feature can be used either to generate a specific frequency, with a good accuracy from any crystal value, or to perform a fine tuning of the frequency on-the-fly.

---

### Table 60. VCO frequency and output frequency in integer mode

<table>
<thead>
<tr>
<th>PLLxVCOSEL = 0</th>
<th>PLLxVCOSEL = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_{VCO} = 2 \times F_{ref_ck} \times (DIVN + 1) )</td>
<td>( F_{VCO} = F_{ref_ck} \times (DIVN + 1) )</td>
</tr>
<tr>
<td>( F_{\text{pll}<em>y_ck} = \frac{F</em>{VCO}}{2 \times (DIVy + 1)} )</td>
<td>( F_{\text{pll}<em>y_ck} = \frac{F</em>{VCO}}{(DIVy + 1)} )</td>
</tr>
</tbody>
</table>

with \( y = P, Q, R, S \) or \( T \)

Refer to section *Section 7.7.1: PLL programming procedure* for additional information.
The VCO frequency (F_{VCO}) and output frequencies expressions are given by the following table:

<table>
<thead>
<tr>
<th>Table 61. VCO frequency and output frequency in fractional mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PLLxVCOSEL = 0</strong></td>
</tr>
<tr>
<td>F_{VCO} = 2 \times \frac{F_{ref_ck} \times (DIVN + 1 + \frac{FRACN}{2^{13}})}{y}</td>
</tr>
<tr>
<td>F_{pll_y_ck} = \frac{F_{VCO}}{2 \times (DIVy + 1)}</td>
</tr>
<tr>
<td>with y = P, Q, R, S or T</td>
</tr>
</tbody>
</table>

Refer to section **Section 7.7.1: PLL programming procedure** for additional information.

**Using the PLLs in spread spectrum mode**

The spread spectrum mode is activated when the SDM loaded with 0, and the bit PLLxSSCGEN = 1. This feature is available on all PLLs.

The VCOH must be selected when the spread spectrum mode is activated. VCOH is selected by setting PLLxVCOSEL to 0. PLLxVCOSEL are located into the **RCC clock configuration register (RCC_CFGR)**.

The spread spectrum technique consist of modulating the VCO frequency with a low-frequency signal (in our case a triangular signal), in order to spread the clock energy into a wider frequency band. As a consequence, the amount of emitted EMI peaks is reduced.

The parameters of the spread spectrum modulation are adjusted via the following fields:

- MODPER[12:0]: used to adjust the modulation frequency
- INCSTEP[14:0]: used to adjust the modulation depth (or modulation index)
- SPREADSEL: used to define if the modulation is centered around the VCO frequency (center-spread) or if it is lowered with respect to VCO frequency (down-spread).

The bits MODPER[12:0], INCSTEP[14:0] and SPREADSEL are located into the RCC_PLLxSSCSR registers with x=1, 2 or 3.

**Figure 49** shows the signal modulating the nominal frequency (F_N), when SPREADSEL = 0 (center-spread) and SPREADSEL = 1 (down-spread). The nominal frequency is the frequency output by the PLL in integer mode, when no clock spreading is applied.

Setting the SPREADSEL bit to ‘1’ (down-spread) guarantees that the PLL output frequency does not exceed the programmed frequency value when the SSCG is enabled.
The peak modulation depth (in percentage) is given by the following formula:

\[ M_D (\%) = \frac{\text{MODPER} \times \text{INCSTEP} \times 100 \times 5}{(2^{15} - 1) \times (\text{DIVN} + 1)} \]

Note that MODPER x INCSTEP must not exceed \((2^{15}-1)\).

The modulation frequency (\(F_{mod}\)) is given by:

\[ F_{mod} = \frac{F_{\text{ck\_ref}}}{4 \times \text{MODPER}} \]

**Note:** Refer to the datasheet of the product for the recommended modulation frequency range.

In order to use the spread spectrum feature, the user has to do the following:

- Program the PLL as explained into **Section : Using the PLLs in integer mode** in order to adjust the nominal frequency (\(F_N\)) according to the targeted by the application.
- Compute the MODPER value according to the wanted modulation frequency (\(F_{\text{mod}}\)):

\[ \text{MODPER} = \text{ROUND} \left( \frac{F_{\text{ck\_ref}}}{4 \times F_{\text{mod}}} \right) \]

- Compute the INCSTEP value according to the wanted modulation depth (\(M_D\)):

\[ \text{INCSTEP} = \text{ROUND} \left( \frac{(2^{15} - 1) \times M_D \times (\text{DIVN} + 1)}{100 \times 5 \times \text{MODPER}} \right) \]

Check that MODPER x INCSTEP does not exceed \((2^{15}-1)\).
- Then DIVP, Q, R, S or T can be adjusted, and the bit PLLxSSCGEN can be set to 1, and the PLL enabled.
The user can check \( F_{\text{MIN}} \), \( F_{\text{MAX}} \), and \( F_C \) as follow:

- If \( \text{SSCG\_MODE} = '0' \) (centered-spread)
  
  \[
  F_C = \frac{F_{\text{PLL}}}{} \times (1 - \frac{M_D}{100})
  \]
  
  \[
  F_{\text{MIN}} = F_C \times (1 - \frac{M_D}{100})
  \]
  
  \[
  F_{\text{MAX}} = F_C \times (1 + \frac{M_D}{100})
  \]

- If \( \text{SSCG\_MODE} = '1' \) (down-spread)
  
  \[
  F_{\text{MAX}} = \frac{F_{\text{PLL}}}{1 - 2 \times \frac{M_D}{100}}
  \]
  
  \[
  F_{\text{MIN}} = F_{\text{MAX}} \times (1 - \frac{M_D}{100})
  \]
  
  \[
  F_C = F_{\text{MAX}} \times (1 - \frac{M_D}{100})
  \]

\text{Figure 50} shows the digital signal generated by Triangular Waveform Generator (TWG block), and the way of MODPER and INCSTEP are changing the triangular waveform.

\text{Refer to section Section 7.7.1: PLL programming procedure for additional information.}

\textbf{7.5.6 System clock (sys\_ck)}

\textbf{System clock selection}

After a system reset, the HSI is selected as system clock and all PLLs are switched OFF. When a clock source is used for the system clock, it is not possible for the software to disable the selected source via the xxxON bits.

Of course, the system clock can be stopped by the hardware when the system enters Stop or Standby mode.

When the system is running, the user application can select the system clock (\texttt{sys\_ck}) among the 4 following sources:

- HSE
- HSI
- CSI
- pll1_p\_ck

This function is controlled by programming the \texttt{RCC clock configuration register (RCC\_CFGR)}. A switch from one clock source to another occurs only if the target clock...
source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready.

The SWS status bits in the RCC clock configuration register (RCC_CFGR) indicate which clock is currently used as system clock. The other status bits in the RCC_CR register indicate which clock(s) is (are) ready.

System clock generation

Figure 51 shows a simplified view of the clock distribution for the CPU and busses. All the dividers shown in the block diagram can be changed on-the-fly without generating timing violations. This feature is a very simple solution to adapt the busses frequencies to the application needs, thus optimizing the power consumption.

The CPRE divider can be used to adjust the CPU clock. However this also impacts the clock frequency of all bus matrix. CPRE is controlled via RCC_CDCFG register.

In the same way, BMPRE divider can be used to adjust the clock for the bus matrix (AHB and AXI), but this also impacts the clock frequency of APB busses. BMPRE is controlled via RCC_BMCFG register.

Most of the prescalers are controlled via RCC_CDCFG and RCC_SRDCFG registers.
Dividers values can be changed on-the-fly. All dividers provide have 50% duty-cycles.

Note that the application must respect the maximum allowed frequencies: \( F_{\text{CPU Max}} \) and \( F_{\text{BUS Max}} \). \( F_{\text{BUS}} \) represents the maximum allowed frequency for the AHB and AXI busses. Refer to the product datasheet for maximum values.

Note as well that the trace clock (\( \text{trace}_{\text{ck}} \)) is generated from \( \text{sys}_{\text{cpu}_{\text{ck}}} \) clock, divided by 3. For additional information refer to Section: Clock distribution for Debug and Trace.
7.5.7 Clock protection

The RCC does not allow to disable the system clock (sys_ck). Several protections are implemented in order to prevent actions disabling accidentally the system clock.

In addition, protection bits are available in order to prevent accidental disabling of FMC and XSPI block activities.

The protections consist on ignoring the invalid write operations. It is recommended to read back registers having protection to insure that the clock configuration expected by the software is accepted by the RCC.

The table hereafter shows the protections handled by the RCC.

<table>
<thead>
<tr>
<th>Protected fields</th>
<th>Conditions checked by the RCC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW[2:0]</td>
<td>Changing SW value is allowed only if the targetted clock is enabled and ready.</td>
</tr>
<tr>
<td>PLLSRC[1:0]</td>
<td>Changing PLLSRC value is allowed if:</td>
</tr>
<tr>
<td></td>
<td>- The targetted clock is enabled and ready, and,</td>
</tr>
<tr>
<td></td>
<td>- All PLL disabled</td>
</tr>
<tr>
<td>HSEON, HSION or CSION</td>
<td>Changing these fields is allowed if:</td>
</tr>
<tr>
<td></td>
<td>- The oscillator is not used directly or indirectly (via PLL1) as system clock, and</td>
</tr>
<tr>
<td></td>
<td>- FMCCKP = XSPICKP = 0.</td>
</tr>
<tr>
<td>DIVMx[5:0], PLLxRGE[1:0], PLLxVCOSEL, DIVN[8:0]</td>
<td>Changing these fields is allowed only if the corresponding PLL is disabled (PLLxON=0)</td>
</tr>
<tr>
<td>DIVP[6:0], DIVQ[6:0], DIVR[6:0], DIVS[2:0], DIVT[2:0]</td>
<td>Changing these fields is allowed only if the corresponding DIVyEN bit is equal to 0 (y = P, Q, R, S or T)</td>
</tr>
<tr>
<td>PLL1ON</td>
<td>Changing this field is allowed if:</td>
</tr>
<tr>
<td></td>
<td>- SW switch is not selecting the PLL1, and</td>
</tr>
<tr>
<td></td>
<td>- FMCCKP = 0.</td>
</tr>
<tr>
<td>PLL1PEN</td>
<td>Disabling the DIVP output of PLL1 is allowed if the SW switch is not selecting PLL1</td>
</tr>
<tr>
<td>PLL2ON</td>
<td>Changing this field is allowed only if FMCCKP = XSPICKP = 0.</td>
</tr>
<tr>
<td>PLL2SEN, PLL2TEN,</td>
<td>Changing these fields is allowed only if XSPICKP = 0.</td>
</tr>
<tr>
<td>XSPI1EN, XSPI2EN,</td>
<td></td>
</tr>
<tr>
<td>XSPI1LPEN XSPI2LPEN, XSPI1RST XSPI2RST</td>
<td></td>
</tr>
<tr>
<td>PLL1QEN, PLL2REN,</td>
<td>Changing these fields is allowed only if FMCCKP = 0.</td>
</tr>
<tr>
<td>FMCEN, FMCLEN, FMCRST</td>
<td></td>
</tr>
<tr>
<td>HSEBYP, HSEEXT</td>
<td>Changing these fields is allowed only if HSEON = 0.</td>
</tr>
<tr>
<td>LSEBYP, LSEEXT</td>
<td>Changing these fields is allowed only if LSEON = 0.</td>
</tr>
</tbody>
</table>

There is also clock protection mechanism in case of HSE failure, refer to Section : CSS on HSE.
Clock generation in Stop and Standby modes

When the whole system enters Stop mode, all the clocks (system and kernel clocks) are stopped and the following clock sources are disabled as well:

- CSI, HSI (depending on HSIKERON, and CSIKERON bits)
- HSE
- PLL1, PLL2 and PLL3
- HS48

The content of the RCC registers is not altered except for PLL1ON, PLL2ON, PLL3ON HSEON and HS48ON that are set to 0.

HSION and CSION bits are also affected depending on STOPWUCK and STOPKERWUCK bits (see Table 63).

When the CPU requested to go in Stop mode, the RCC first stop all the requested clocks, and inform the PWR that all clocks have been properly stopped. As shown in Figure 52, three main signals are used to control power transitions:

- The rcc_pwrds is used to indicate to the PWR that the RCC has stopped all the clocks, and thus the PWR can go to Stop or Standby.
- The pwr_wkup is used to indicate to the RCC to re-enable the clocks.
- The exti_wkup is used to indicate to the PWR that an event requests to exit the system from Stop mode.

Exiting Stop mode

When the microcontroller exits system Stop mode via a wakeup event, the application can select which oscillator (HSI and/or CSI) is used to restart. The STOPWUCK bit selects the oscillator used as system clock. The STOPKERWUCK bit selects the oscillator used as kernel clock for peripherals. The STOPKERWUCK bit is useful if after a system Stop, a peripheral needs a kernel clock generated by an oscillator different from the one used for the system clock.

All these bits belong to the RCC clock configuration register (RCC_CFGR).

Table 63 gives a detailed description of their behavior.
During Stop mode

There are two specific cases where the HSI or CSI can be enabled during system Stop mode:

- when a dedicated peripheral requests the kernel clock
  The peripheral receives the HSI or CSI according to the kernel clock source selected for this peripheral (via PERxSRC).
- when the HSIKERON or CSIKERON bits are set the HSI and CSI are kept running during Stop mode but the outputs are gated. The clock is then available immediately when the system exits Stop mode or when a peripheral requests the kernel clock (see Table 64 for details).

HSIKERON and CSIKERON bits belong to RCC register map. Table 64 gives a detailed description of their behavior.

Table 64. HSIKERON and CSIKERON behavior

<table>
<thead>
<tr>
<th>HSIKERON (CSIKERON)</th>
<th>-</th>
<th>HSI (CSI) state during Stop mode</th>
<th>HSI (CSI) setting time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>→</td>
<td>OFF</td>
<td>( t_{su(HSI)} ) ( t_{su(CSI)} ) (^{(1)})</td>
</tr>
<tr>
<td>1</td>
<td>→</td>
<td>Running and gated</td>
<td>Immediate</td>
</tr>
</tbody>
</table>

1. \( t_{su(HSI)} \) and \( t_{su(CSI)} \) are the startup times of the HSI and CSI oscillators (refer to the product datasheet for values of these parameters).

When the microcontroller exists Standby mode, the HSI is selected as system and kernel clock. The RCC registers are reset to their initial values except for the RCC_RSR and RCC_BDCR registers.

Note that the HSI and CSI outputs provide two clock paths (see Figure 40):

- one path for the system clock (hsi_ck or csi_ck)
- one path for the peripheral kernel clock (hsi_ker_ck or csi_ker_ck).

When a peripheral requests the kernel clock in system Stop mode, only the path providing the hsi_ker_ck or csi_ker_ck is activated.

Caution: The CPU does not get automatically the same clock frequencies when leaving Stop mode: it is up to the application to restore the previous clock settings if needed.
### 7.5.9 Peripheral clock distribution

Some peripherals are designed to work with two different clock domains that operate asynchronously:

- a clock domain synchronous with the register and bus interface (`ckg_bus_perx` clock)
- a clock domain generally synchronous with the peripheral (kernel clock)

The benefit of having peripherals supporting these two clock domains is that the user application has more freedom to choose optimized clock frequency for the CPU, bus matrix and for the kernel part of the peripheral.

As a consequence, the user application can change the bus frequency without reprogramming the peripherals. As an example an on-going transfer with UART is not disturbed if its APB clock is changed on-the-fly.

*Table 65* shows the clocks the RCC delivers to the peripherals. Note as well that the clock named `per_ck`, is the output of a mux which allows the application to select:

- `hsi_ker_ck`, or
- `csi_ker_ck`, or
- `hse_ker_ck`

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock types</th>
<th>Clock sources</th>
<th>Kernel clock MUX</th>
<th>Max Kernel clock freq. [MHz]</th>
<th>Type (2/3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pos.</td>
<td>Control field</td>
<td></td>
</tr>
<tr>
<td>ADF</td>
<td>Kernel</td>
<td>hclk1</td>
<td>0**(4)**</td>
<td>ADF1SEL</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_p_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2S_CKIN</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>csi_ker_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>hclk1</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
<td>-</td>
</tr>
<tr>
<td>ADC12</td>
<td>Kernel</td>
<td>pll2_p_ck</td>
<td>0**(4)**</td>
<td>ADCSEL</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_r_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>hclk1</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
<td>-</td>
</tr>
<tr>
<td>CORDIC</td>
<td>Bus</td>
<td>hclk2</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
</tr>
<tr>
<td>CRC</td>
<td>Bus</td>
<td>hclk4</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
</tr>
<tr>
<td>CRS</td>
<td>Bus</td>
<td>pclk1</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
</tr>
<tr>
<td>CRYP</td>
<td>Bus</td>
<td>hclk3</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
</tr>
<tr>
<td>DBG</td>
<td>Bus</td>
<td>sys_bus_ck</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sys_cpu_ck/3</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 3</td>
</tr>
<tr>
<td>DB_OCSP1.</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
</tr>
<tr>
<td>DB_OCSP2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Peripherals</td>
<td>Clock types</td>
<td>Clock sources</td>
<td>Kernel clock MUX</td>
<td>Max Kernel clock freq. [MHz]</td>
<td>Type</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>---------------</td>
<td>------------------</td>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pos.</td>
<td>Control field</td>
<td>(1)</td>
</tr>
<tr>
<td>DB_SDMMC1</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>DB_SDMMC2</td>
<td>Bus</td>
<td>hclk2</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>DCMIPP</td>
<td>Bus</td>
<td>aclk</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pclk5</td>
<td></td>
<td></td>
<td>$F_{\text{MAX}} / 4$</td>
</tr>
<tr>
<td>GPDMA1</td>
<td>Bus</td>
<td>hclk1</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>HPDMA1</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>DMA2D</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>ETH1</td>
<td>Kernel</td>
<td>ETH_MII_TX_CLK</td>
<td>-</td>
<td>-</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETH_MII_TX_CLK/.ETH_RMII_REF_CLK</td>
<td>0</td>
<td>ETH1REFCKSEL</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck</td>
<td>1</td>
<td></td>
<td>ETH1PHYCKSEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>eth_clk_fb</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck</td>
<td>0</td>
<td>ETH1PHYCKSEL</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_s_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>clk_ptp_ref_i</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk1</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>EXTI</td>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 4$</td>
</tr>
<tr>
<td>FLASH</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>FDCAN</td>
<td>Kernel</td>
<td>hse_ker_ck</td>
<td>0</td>
<td>FDCANSEL</td>
<td>125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll1_q_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk1</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 4$</td>
</tr>
<tr>
<td>FMC</td>
<td>Kernel</td>
<td>hclk5</td>
<td>0</td>
<td>FMCSEL</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll1_q_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_r_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aclk</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GPIOA-H. GPIOM-P</td>
<td>Bus</td>
<td>hclk4</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
<tr>
<td>GPU2D</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>$F_{\text{MAX}} / 2$</td>
</tr>
</tbody>
</table>
### Table 65. Peripheral clock distribution summary

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock types</th>
<th>Clock sources</th>
<th>Kernel clock MUX</th>
<th>Max Kernel clock freq. [MHz] (1)</th>
<th>Type (2)(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFXMMU</td>
<td>Bus</td>
<td>hclk5, aclk</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
<td>-</td>
</tr>
<tr>
<td>GFXTIM</td>
<td>Bus</td>
<td>pclk5</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>-</td>
</tr>
<tr>
<td>HASH</td>
<td>Bus</td>
<td>hclk3</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
<td>-</td>
</tr>
<tr>
<td>HDMI-CEC</td>
<td>Kernel</td>
<td>lse_ck, lsi_ck, csi_ker_ck/122</td>
<td>CECSEL 1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Bus</td>
<td>pclk1</td>
<td></td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>-</td>
</tr>
<tr>
<td>I2C2, I2C3</td>
<td>Kernel</td>
<td>pclk1, pll3_r_ck, hsi_ker_ck, csi_ker_ck</td>
<td>I2C23SEL</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>A</td>
</tr>
<tr>
<td>Bus</td>
<td>pclk1</td>
<td></td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>-</td>
</tr>
<tr>
<td>I2C1/I3C1</td>
<td>Kernel</td>
<td>pclk1, pll3_r_ck, hsi_ker_ck, csi_ker_ck</td>
<td>I2C1_I3C1SEL</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>A</td>
</tr>
<tr>
<td>Bus</td>
<td>pclk1</td>
<td></td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>-</td>
</tr>
<tr>
<td>XSPIM</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
<td>-</td>
</tr>
<tr>
<td>IWDG</td>
<td>Kernel</td>
<td>lsi_ck</td>
<td>-</td>
<td>1</td>
<td>A</td>
</tr>
<tr>
<td>Bus</td>
<td>pclk4</td>
<td></td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>-</td>
</tr>
<tr>
<td>JPEG</td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 2</td>
<td>-</td>
</tr>
<tr>
<td>LPTIM1</td>
<td>Kernel</td>
<td>pclk1, pll2_p_ck, pll3_r_ck, lse_ck, lsi_ck, per_ck</td>
<td>LPTIM1SEL</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>A</td>
</tr>
<tr>
<td>Bus</td>
<td>pclk1</td>
<td></td>
<td>-</td>
<td>F&lt;sub&gt;MAX&lt;/sub&gt; / 4</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 65. Peripheral clock distribution summary

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock types</th>
<th>Clock sources</th>
<th>Kernel clock MUX</th>
<th>Max Kernel clock freq. [MHz] (1)</th>
<th>Type (2)/(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pos.</td>
<td>Control field</td>
<td></td>
</tr>
<tr>
<td>LPTIM2, LPTIM3</td>
<td>Kernel</td>
<td>pclk4</td>
<td>0(^{(4)})</td>
<td>LPTIM23SEL</td>
<td>F(_{MAX}/4) A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_r_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lse_ck</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lsi_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/4) -</td>
</tr>
<tr>
<td>LPTIM4, LPTIM5</td>
<td>Kernel</td>
<td>pclk4</td>
<td>0(^{(4)})</td>
<td>LPTIM45SEL</td>
<td>F(_{MAX}/4) A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_r_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lse_ck</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lsi_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/4) -</td>
</tr>
<tr>
<td>LPUART1</td>
<td>Kernel</td>
<td>pclk4</td>
<td>0(^{(4)})</td>
<td>LPUART1SEL</td>
<td>F(_{MAX}/4) A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_q_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_q_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>csi_ker_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lse_ck</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/4) -</td>
</tr>
<tr>
<td>LTDC</td>
<td>Kernel</td>
<td>pll3_r_ck</td>
<td>-</td>
<td></td>
<td>90 A</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk5</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/4) -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aclk</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/2) -</td>
</tr>
<tr>
<td>MCE1, MCE2, MCE3</td>
<td>Bus</td>
<td>aclk</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/2) -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>MDIO</td>
<td>Bus</td>
<td>pclk1</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/4) -</td>
</tr>
<tr>
<td>PKA</td>
<td>Bus</td>
<td>hclk3</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/2) -</td>
</tr>
<tr>
<td>PWR</td>
<td>Bus</td>
<td>hclk4</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/2) -</td>
</tr>
<tr>
<td>PSSI</td>
<td>Kernel</td>
<td>pll3_r_ck</td>
<td>0(^{(4)})</td>
<td>PSSI Sel</td>
<td>100 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk2</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX}/2) -</td>
</tr>
</tbody>
</table>
### Table 65. Peripheral clock distribution summary

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock types</th>
<th>Clock sources</th>
<th>Kernel clock MUX</th>
<th>Max Kernel clock freq. [MHz] (1)</th>
<th>Type (2)(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pos.</td>
<td>Control field</td>
<td></td>
</tr>
<tr>
<td>XSPI1</td>
<td>Kernel</td>
<td>hclk5</td>
<td>0(4)</td>
<td>XSPI1SEL</td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_s_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_t_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aclk</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XSPI2</td>
<td>Kernel</td>
<td>hclk5</td>
<td>0(4)</td>
<td>XSPI2SEL</td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_s_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_t_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aclk</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTGFS</td>
<td>Kernel</td>
<td>hsi48_ker_ck</td>
<td>0(4)</td>
<td>OTGFSSEL</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_q_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>clk48mohci</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk1</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td>OTGHS</td>
<td>Kernel</td>
<td>phy60m_ck</td>
<td>-</td>
<td></td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk1</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td>RCC</td>
<td>Bus</td>
<td>hclk4</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td>RNG</td>
<td>Kernel</td>
<td>hsi48_ker_ck</td>
<td>-</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk3</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td>RTC/AWU(5)</td>
<td>Kernel</td>
<td>no clock</td>
<td>0(4)</td>
<td>RTCSEL</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lse_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lsi_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck / (RTCDIV+1)</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td></td>
<td>F_MAX / 4</td>
</tr>
<tr>
<td>SAES</td>
<td>Kernel</td>
<td>hclk3</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>hclk3</td>
<td>-</td>
<td></td>
<td>F_MAX / 2</td>
</tr>
<tr>
<td>SAI1</td>
<td>Kernel</td>
<td>pll1_q_ck</td>
<td>0(4)</td>
<td>SAI1SEL</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_p_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>l2S_CKIN</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_cck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk2</td>
<td>-</td>
<td></td>
<td>F_MAX / 4</td>
</tr>
</tbody>
</table>
## Table 65. Peripheral clock distribution summary

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock types</th>
<th>Clock sources</th>
<th>Kernel clock MUX</th>
<th>Max Kernel clock freq. (MHz) (1)</th>
<th>Type (2)(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI2</td>
<td>Kernel</td>
<td>pll1_q_ck</td>
<td>0(^{(4)})</td>
<td>SAI2SEL</td>
<td>133</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_p_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>l2S_CKIN</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>spdif_symb_ck</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>pclk2</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX})/4</td>
<td>-</td>
</tr>
<tr>
<td>SBS</td>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td>F(_{MAX})/4</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC1</td>
<td>Kernel</td>
<td>pll2_s_ck</td>
<td>0(^{(4)})</td>
<td>SDMMC12SEL</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_t_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>hclk5</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX})/2</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC2</td>
<td>Kernel</td>
<td>pll2_s_ck</td>
<td>0(^{(4)})</td>
<td>SDMMC12SEL</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_t_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>hclk2</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX})/2</td>
<td>-</td>
</tr>
<tr>
<td>SPDIFRX</td>
<td>Kernel</td>
<td>pll1_q_ck</td>
<td>0(^{(4)})</td>
<td>SPDI4RXSEL</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_r_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_r_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>pclk1</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX})/4</td>
<td>-</td>
</tr>
<tr>
<td>SPI/12S6</td>
<td>Kernel</td>
<td>pclk4</td>
<td>0(^{(4)})</td>
<td>SPI6SEL</td>
<td>200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_q_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_q_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>csi_ker_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>pclk4</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX})/4</td>
<td>-</td>
</tr>
<tr>
<td>SPI/12S1</td>
<td>Kernel</td>
<td>pll1_q_ck</td>
<td>0(^{(4)})</td>
<td>SPI1SEL</td>
<td>130</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_p_ck</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>l2S_CKIN</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bus</td>
<td>pclk2</td>
<td>-</td>
<td>-</td>
<td>F(_{MAX})/4</td>
<td>-</td>
</tr>
</tbody>
</table>
### Table 65. Peripheral clock distribution summary

<table>
<thead>
<tr>
<th>Peripherals</th>
<th>Clock types</th>
<th>Clock sources</th>
<th>Kernel clock MUX</th>
<th>Max Kernel clock freq. [MHz] (1)</th>
<th>Type (2)(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI4, SPI5</td>
<td>Kernel</td>
<td>pclk2</td>
<td>SPI45SEL</td>
<td>200</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_q_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_q_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>csi_ker_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk2</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPI/I2S3, SPI/I2S2</td>
<td>Kernel</td>
<td>pclk1</td>
<td>SPI23SEL</td>
<td>200</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll1_q_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_p_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_p_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2S_CKIN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>per_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk1</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TAMPER</td>
<td>Kernel</td>
<td>no clock</td>
<td>RTCSEL</td>
<td>4</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lse_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lsi_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hse_ker_ck / (RTCDIV+1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk4</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DTS</td>
<td>Kernel</td>
<td>lse_ck</td>
<td></td>
<td>10</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk4</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, TIM12, TIM13, TIM14</td>
<td>Kernel</td>
<td>timg1_ck</td>
<td></td>
<td>FMAX / 2</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk1</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM1, TIM9, TIM15, TIM16, TIM17</td>
<td>Kernel</td>
<td>timg2_ck</td>
<td></td>
<td>FMAX / 2</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk2</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USART1,</td>
<td>Kernel</td>
<td>pclk2</td>
<td>USART1SEL</td>
<td>FMAX / 4</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll2_q_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pll3_q_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hsi_ker_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>csi_ker_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>lse_ck</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bus</td>
<td>pclk2</td>
<td></td>
<td>FMAX / 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Figure 53 to Figure 68 provide a more detailed description of kernel clock distribution. Refer to Section 7.5.12: Peripheral clock gating control for more details.

To reduce the amount of switches, some peripherals share the same kernel clock source. Nevertheless, all peripherals have their dedicated enable signal.

**Clock distribution for ADF, SAIs, and SPDIFRX**

The audio peripherals generally need specific accurate frequencies, except for SPDIFRX. As shown in Figure 53 and Figure 56, the kernel clock of the SAIs or SPI(I2S)s can be generated by:

- PLL1 when the amount of active PLLs must be reduced (for SAIs and SPI/I2S1 to 3)
- APB2 peripheral clock (for SPI/I2S4 and 5)
- APB4 peripheral clock (for SPI/I2S6)
- PLL2 or 3 for optimal flexibility in frequency generation
- HSE, HSI or CSI for use-cases where the current consumption is critical
- I2S_CKIN when an external clock reference needs to be used
The SPDIFRX does not require a specific kernel clock frequency but only a frequency high enough to sample properly the incoming stream. Refer to the SPDIFRX description for more details.

The SPDIFRX also provides a symbol clock (spdifrx_symb_ck) which is 64 times faster than the received samples. The symbol clock is proposed as kernel clock for SAI2 in order to ease bridging from SPDIFRX to SAI2.

The audio blocks also have common kernel clocks in order to ease data exchange.

The ADF can work even in Stop mode as long as the bus clock is not requested. Working in Stop mode is possible only if the application selected csi_ker_ck or hsi_ker_ck as kernel clock sources. The ADF asserts the adf_ker_ckreq when the kernel clock is requested.

The kernel clock mechanism works only if the kernel clock source is provided by an RC oscillator (hsi_ker_ck or csi_ker_ck).
1. **X** represents the selected mux input after a system reset.

**Clock distribution for SPIs, SPI/I2Ss**

SPI peripherals do not need an accurate kernel clock frequency but a clock fast enough for the serial interface. The SPI/I2S peripherals may request accurate kernel clock frequency when using the I2S function. For that purpose the source can be selected among the following ones:

- The bus clock (for SPIs)
- PLL1 when the amount of active PLLs must be reduced
- PLL2 or PLL3 if better flexibility is required. As an example, this solution allows changing the frequency bus via PLL1 without affecting the speed of some serial interfaces.
- HSI or CSI or per_ck for low-power use-cases.
The SPI/I2S also have common kernel clocks with audio blocks in order to ease data exchange.

Figure 54. Clock distribution for SPIs and SPI/I2S

1. X represents the selected switch input after a system reset.

Clock distribution for I2C[3:2], and I2C1/I3C1

I2Cs and I3C1 peripherals do not need an accurate kernel clock frequency but a clock fast enough to handle properly the serial interface. The following kernel clock sources are proposed:

- The bus clock
- PLL3 in the case were the bus clock cannot be used
- HSI or CSI for low-power use-cases

The I2Cs and I3C1 can work even in Stop mode as long as the bus clock is not requested. Working in Stop mode is possible only if the application selected csi_ker_ck or hsi_ker_ck as kernel clock sources. The I2Cs and I3C1 assert the i2c_ker_ckreq or i3c_ker_ckreq signals, when the kernel clock is requested.
The kernel clock mechanism works only if the kernel clock source is provided by an RC oscillator (hsi_ker_ck or csi_ker_ck).

Figure 55. Clock distribution for I2C[3:2] and I2C1/I3C1

1. X represents the selected switch input after a system reset
Clock distribution for UARTs and USARTs

UARTs need kernel clock frequency allowing the adjustment of the baud rate with an acceptable accuracy (generally less than +/−2%). For that purpose the source can be selected among the following ones:

- The bus clock
- PLL3 in the case were the bus clock cannot be used
- HSI, CSI or LSE for low-power use-cases

The UARTs can work even in Stop mode as long as the bus clock is not requested. Working in Stop mode is possible only if the application selected csi_ker_ck or hsi_ker_ck as kernel clock sources. The UARTs assert the uartx_ker_ckreq signal, when the kernel clock is requested. The kernel clock mechanism works only if the kernel clock source is provided by an RC oscillator (hsi_ker_ck or csi_ker_ck).

UARTs can also work in Stop mode using the LSE clock when high baud rates are not required.

Figure 56. Clock distribution for UARTs, LPUART1 and USARTs

1. X represents the selected switch input after a system reset.
Clock distribution for FDCAN

The switch is dynamic: the transition between two inputs is glitch-free.

1. X represents the selected switch input after a system reset.
Clock distribution for graphic blocks (GPU2D, LTDC, DCMIPP and PSSI)

Figure 58. Clock distribution for GPU2D

1. X represents the selected switch input after a system reset.

Figure 59. Clock distribution for LTDC and DCMIPP

1. X represents the selected switch input after a system reset.

The PSSI receives an AHB clock and a kernel clock (pxclk).

The pxclk can be provided either by an external device via PSSI_PIXCK pad, or by the RCC. Note that the clock generated by the RCC is provided to pxclk input by the feedback path of the PSSI_PIXCK pad. The drive of the PSSI_PIXCK is controlled by the PSSI block.
Clock distribution for SDMMCs, FMC, and XSPIs

The FMC kernel clock can be chosen between 4 different sources, giving good flexibility.

For each XSPI, a clock switch allows the selection between 3 different sources, giving good flexibility. It is possible to enable independently each XSPI block.

The switches FMCSEL, XSPI[2:1]SEL, also embeds several protections:

- In case of HSE failure detected by the HSECSS function, the XSPI[2:1]SEL switches and the FMCSEL switch go to recovery position in order to provide a default clock.
- A protection bit: FMCKP prevents the application to accidentally change the FMCSEL value. Refer to Section 7.5.7: Clock protection for details.
- A protection bit: XSPICKP prevents the application to accidentally change the XSPI[2:1]SEL values. Refer to Section 7.5.7: Clock protection for details.
- Status fields (FMCSWP, XSPI[2:1]SWP) located into RCC clock protection register (RCC_CKPROTR) allows the application to check the effective position of the switches.
- If one of the switch is selecting an input with an invalid clock, the switch outputs a recovery clock preventing a memory crash and allowing the application to log or correct the issue. The RCC provides hclk5/4 as recovery clock. In order to work fine, the
application must ensure that the recovery clock frequency is supported by the external device.

- For the XSPI[2:1]SEL switches:
  - Going to position 1 is allowed if PLL2ON = PLL2SEN = 1. If the application switches to position 1 when these conditions are not met, the switch generates the recovery clock.
  - Going to position 2 is allowed if PLL2ON = PLL2TEN = 1. If the application switches to position 2 when these conditions are not met, the switch generates the recovery clock.

- For the FMCSEL switch:
  - Going to position 1 is allowed if PLL1ON = PLL1QEN = 1. If the application switches to position 1 when these conditions are not met, the switch generates the recovery clock.
  - Going to position 2 is allowed if PLL2ON = PLL2REN = 1. If the application switches to position 2 when these conditions are not met, the switch generates the recovery clock.
  - Going to position 3 is allowed if HSION = 1. If the application switches to position 3 when this condition is not met, the switch generates the recovery clock.

How to handle properly XSPI switches:

- Switch on the wanted clock source,
- Ensure the clock source is ready, and conditions described above are met,
- Set XSPI1SEL to the wanted position,
- Enable the XSPI clock (XSPIxEN = 1),
- Read XSPIISWP field from RCC_CKPROTR register,
- If the position given by XSPIISWP is the same as the wanted, it means that the XSPIx kernel clock is as expected.
- If the position given by XSPIISWP is 000, it means that there is no clock present at the selected input.

The XSPIs must provide a clock to the external memory, with a duty cycle distortion generally lower than 5%. In order to reach this requirement, the kernel clock provided to the XSPIs has a typical duty cycle of 50%. In addition, the XSPIs embed prescaler allowing clock division by even ratios.
1. \( \times \) represents the selected switch input after a system reset.

The SDMMC1 and SDMMC2 share the same kernel clock. A clock switch allows the selection between 2 different sources. It is possible to enable independently each SDMMC block. Note that when one of the SDMMC is enabled via its SDMMC[2:1]EN bit, the associated delay block is enabled as well.

The application must take care about the duty-cycle of the kernel clock provided to the SDMMC blocks.

### Table 66. SDMMC interface clock constraints

<table>
<thead>
<tr>
<th>SDMMC mode</th>
<th>Mode name</th>
<th>Interface clock frequency</th>
<th>Duty cycle constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDIO</td>
<td>SDR12</td>
<td>25 MHz or less</td>
<td>30 - 70%</td>
</tr>
<tr>
<td></td>
<td>SDR25</td>
<td>50 MHz or less</td>
<td>30 - 70%</td>
</tr>
<tr>
<td></td>
<td>DDR50</td>
<td>50 MHz or less</td>
<td>45 - 55%</td>
</tr>
<tr>
<td></td>
<td>SDR50</td>
<td>100 MHz or less</td>
<td>30 - 70%</td>
</tr>
</tbody>
</table>
For example, if the SDMMC works in SDR50, then a kernel clock of 50 MHz, with a duty cycle better than 30-70% is enough. If the SDMMC works in DDR50, then it is recommended to provide a kernel clock of 100 MHz, and divide the frequency of the kernel clock by two, using the SDMMC divider in order to insure a duty-cycle very close to 50% for the SDMMC_CK.

Table 66. SDMMC interface clock constraints

<table>
<thead>
<tr>
<th>SDMMC mode</th>
<th>Mode name</th>
<th>Interface clock frequency</th>
<th>Duty cycle constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.MMC</td>
<td>Backward compatible</td>
<td>26 MHz or less</td>
<td>30 - 70%</td>
</tr>
<tr>
<td></td>
<td>High Speed SDR</td>
<td>52 MHz or less</td>
<td>30 - 70%</td>
</tr>
<tr>
<td></td>
<td>High Speed DDR</td>
<td>52 MHz or less</td>
<td>45 - 55%</td>
</tr>
</tbody>
</table>

Figure 62. Clock distribution for SDMMC[2:1] and DB_SDMMC[2:1]

1. X represents the selected switch input after a system reset.
Clock distribution for OTGFS, OTGHS and UCPD

*Figure 63* shows the clock distribution for the USB blocks.

- The USB type-C power delivery block (UCPD), uses ucpd_ker_ck as kernel clock. The ucpd_ker_ck is directly generated from the HSI output divided by 4.
- The USBPHYC, embeds a PLL accepting a reference input frequency of 16, 19.2, 20, 24, 26 or 32 MHz. The reference clock can be selected among:
  - hse.osc_ck
  - hse.ker_ck/2 or
  - pll3_q_ck

  The USBPHYC provides a 60 MHz clock to the OTGHS block, and a 48 MHz clock to the OTGFS block. The clock hse.osc_ck is the direct output of the HSE oscillator.

- The OTGHS receives the 60 MHz clock from the USBPHYC (phy60m_ck), when working in HS or FS mode.
- The OTGFS receives a 48 MHz clock. The application can select:
  - the clk48mohci (from USBPHYC)
  - the hsi48_ker_ck for crystal-less applications
  - hse.ker_ck or
  - pll3_q_ck

The selection of the reference clock for the USBPHYC is performed by a simple MUX. In order to change the clock source, the application must follow the sequence hereafter:

- Disable the USBPHYC clock by setting USBPHYCEN to 0
- Change the clock source selector (USBPHYCSEL) to the wanted value
- Enable the USBPHYC clock by setting USBPHYCEN to 1.
Figure 63. Clock distribution for USB and UCPD

1. \( \times \) represents the selected switch input after a system reset.
Clock distribution for ETH1

The Ethernet clocks provided by the RCC are available on ETH_CLK pad. The application can select if the ETH_CLK is generated from the HSE kernel clock or from the pll3_s_ck.

The RCC also provides the bus clock and the reference clock for the PTP function.

Note that the bus and PTP clocks generation are controlled via ETH1MACEN and ETH1MACLPEN bits.

Figure 64. Clock distribution for ETH1

1. \( X \) represents the selected switch input after a system reset.

The signal ETH_SEL is provided by the SBS block and defines the interface type used:

In MII mode (orange path):
- The transmit clock is received from the external PHY via the pad ETH_MII_TX_CLK.
- The receive clock is received from the external PHY via the pad ETH_MII_RX_CLK.
- The clock frequency is 2.5 MHz for Ethernet at 10 Mbps or 25 MHz for Ethernet at 100 Mbps.
In RMII mode (blue path):

- The reference clock (50 MHz) can be selected between several sources:
  - The ETH_RMII_REF_CLK pad, if the reference clock is provided by the PHY
  - The hse_ker_clk in case the Ethernet PHY is providing a reference clock connected to OSC_IN
  - The ETH_CLK feedback clock (eth_clk_fb) if the RCC is providing the reference clock to the PHY. This selection is controlled via ETH1REFCKSEL[1:0].

- The ETH_SEL must be set to 1,
  - a clock must be provided to clk_rx_i and clk_tx_i inputs
  - The clock frequency at clk_rx_i and clk_tx_i inputs is 2.5 MHz (division by 20) for Ethernet at 10 Mbps, and 25 MHz (division by 2) for Ethernet at 100 Mbps.
  - The transmit and receive clock frequency can be adjusted dynamically to 2.5 or 25 MHz according to the signal mac_speed_o[0] provided by the ETH block.

Note that the ETH1REFCKSEL mux position cannot be changed when a clock is provided to the ETH1 block. In order to select the wanted mux position the following sequence must be respected:

- Set bits ETH1TXEN, ETH1RXEN and ETH1MACEN to 0,
- Set ETH1REFCKSEL to the wanted position,
- Set bits ETH1TXEN, ETH1RXEN and ETH1MACEN to 1

The RCC can generate a reference clock of 25 or 50 MHz to the external PHY, via ETH_CLK pad. The figure hereafter shows the possible clock configurations.

The ETH_CLK is generated only if:

- ETH1TXEN or ETH1RXEN are enabled, and
- the system is in Run or Sleep, and
- the clock source selected by ETH1PHYCKSEL is available.

The bits ETH1REFCKSEL[1:0] and ETH1PHYCKSEL are located into the RCC AHB peripheral kernel clock selection register (RCC_CCIPR1).

**Figure 65. ETH clock configuration**

![Figure 65. ETH clock configuration](image)

The ETH1 block provides information on the MAC speed (mac_speed_o[1:0]), which may change dynamically:

- mac_speed_o[1:0] = 0b10: for 10 Mbps
- mac_speed_o[1:0] = 0b11 or 0b0x: for 100 Mbps
Clock distribution for ADC1-2

Figure 66. Clock distribution For ADCs

1. \( X \) represents the selected switch input after a system reset.

Clock distribution for SAES and RNG

Figure 67. Clock distribution for SAES and RNG

1. \( X \) represents the selected switch input after a system reset.

The RNG block is using the hsi48_ker_ck as kernel clock. The logic handling controlling the clock of the RNG is different from standard peripherals. When enabled, the RNG has to assert its kernel clock request, in order to get a kernel clock. For example, if the SAES needs the RNG resource, the RNG must also be enabled. Refer to Section 7.5.12: Peripheral clock gating control for details.
Clock distribution for HDMI-CEC

Figure 68. Clock distribution for HDMI-CEC

1. X represents the selected switch input after a system reset

Clock distribution for TIM and LPTIM

The TIMs timers are using kernel clocks (tim_ker_ck) phase aligned with the bus interface clock.

The working frequency of the timers clock depends on:
- The APB prescaler corresponding to the bus where the timer is connected, and
- The bit TIMPRE

Figure 69. Clock distribution for TIMs

1. X represents the selected switch input after a system reset
2. Can be changed on the fly
The Table 67 shows how to select the timer clock frequency.

### Table 67. Ratio between clock timer and pclk

<table>
<thead>
<tr>
<th>PPRE1(1) PPRE2</th>
<th>TIMPRE (2)</th>
<th>F&lt;sub&gt;rcc_timx_ker_ck&lt;/sub&gt;</th>
<th>F&lt;sub&gt;rcc_timy_ker_ck&lt;/sub&gt;</th>
<th>F&lt;sub&gt;pclk1&lt;/sub&gt;</th>
<th>F&lt;sub&gt;pclk2&lt;/sub&gt;</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xx x</td>
<td>x</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt;</td>
<td></td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt;</td>
<td></td>
<td>The timer clock is equal to the bus clock.</td>
</tr>
<tr>
<td>100 x</td>
<td>x</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt;</td>
<td></td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 2</td>
<td></td>
<td>The timer clock is twice as fast as the bus clock.</td>
</tr>
<tr>
<td>101 0</td>
<td>0</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 2</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 8</td>
<td></td>
<td></td>
<td>The timer clock is 4 times faster than the bus clock.</td>
</tr>
<tr>
<td>110 0</td>
<td>0</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111 0</td>
<td>0</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 8</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101 1</td>
<td>1</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt;</td>
<td></td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110 1</td>
<td>1</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 4</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>111 1</td>
<td>1</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 4</td>
<td>F&lt;sub&gt;hclk1&lt;/sub&gt; / 16</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. PPRE1 and PPRE2 are the prescaler for the APB1 and APB2 clocks.
2. TIMPRE belongs to RCC clock configuration register (RCC_CFGR).

The LPTIMs timers have a kernel clocks fully asynchronous with respect to their bus interface clock. The kernel clock can be selected among up to 6 clock sources.

The LPTIMs also use lse_ck and lsi_ck as kernel clock, allowing them to work even when the system is in Stop mode.
Figure 70. Clock distribution for LPTIMs

The switch is dynamic: the transition between two inputs is glitch-free.

The **rtc_ck** clock source can be one of the following:

- the **hse_rtc_ck** (**hse_ker_ck** divided by a programmable prescaler)
- the **lse_ck**
- the **lsi_ck** clock

The source clock is selected by programming the RTCSEL[1:0] bits in the **RCC Backup domain control register (RCC_BDCR)** and the RTCPRE[5:0] bits in the **RCC clock configuration register (RCC_CFGR)**.

This selection cannot be modified without resetting the Backup domain.
1. X represents the selected switch input after a system reset

If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_DD supply disappears.

The LSE clock is in the Backup domain, whereas the other oscillators are not. As a consequence:

- If LSE is selected as RTC clock, the RTC continues working even if the V_DD supply is switched OFF, provided the V_BAT supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_DD supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_DD supply is powered off or if the V_CORE supply is powered off. In addition the HSE is not available if the system goes to Stop.

The rtc_ck clock is enabled through RTCEN bit located in the RCC Backup domain control register (RCC_BDCR).

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN and RTCAPBLPEN bits located in RCC_APB4ENR/LPENR registers.

**Note:** To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency (F_APB < 7 x F_RTCLOCK), the software must read the calendar time and date registers twice. The data are correct if the second read access to RTC_TR gives the same result than the first one. Otherwise a third read access must be performed.

**Clock distribution for watchdog**

The RCC provides the clock for the two watchdog blocks available on the circuit. The independent watchdog (IWDG) is connected to the LSI. The window watchdog (WWWDG) is connected to the APB clock.

If an independent watchdog is started by either hardware option or software access, the LSI is forced ON and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

The window watchdog clock (pclk1) can be enabled by setting the WWWDGEN bit in RCC_APB1ENR register. The software cannot stop WWWDG down-counting by setting WWWDGEN bit to ‘0’.
The WWDG block is frozen when the MCU goes to Stop mode.

**Figure 72. Clock distribution for WWDG and IWDG**

![Clock distribution diagram](image)

1. X represents the selected switch input after a system reset

**Clock distribution for Debug and Trace**

The clock generation for the trace and debug is controlled by the DBGMCU block. The DBGCKEN bit allows the application to provide a clock to the debug components. It is also possible to enable this clock via the Debug Access Port.

The trace clock generation is controlled via TRACECKEN bit.

**Figure 73. Clock distribution for DBG and trace**

![Clock distribution diagram](image)

1. X represents the selected switch input after a system reset

**Clock frequency measurement using TIMx**

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

- Calibrating the HSI or CSI with the LSE:

The primary purpose of having the LSE connected to a TIMx input capture is to be able to accurately measure the HSI or CSI. This requires to use the HSI or CSI as system clock source either directly or via PLL1. The number of system clock counts between...
consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm) we can determine the internal clock frequency with the same resolution, and trim the source to compensate for manufacturing-process and/or temperature- and voltage-related frequency deviations.

The basic concept consists in providing a relative measurement (e.g. HSI/LSE ratio). The precision is therefore tightly linked to the ratio between the two clock sources. The greater the ratio is, the more accurate the measurement is.

The HSI and CSI oscillators have dedicated user-accessible calibration bits for this purpose (see **RCC CSI calibration register (RCC_CSICFGR)**). When HSI or CSI is used via the PLLx, the system clock can also be fine-tuned by using the fractional divider of the PLLs.

- Calibrating the LSI with the HSI:

  The LSI frequency can also be measured: this is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process deviation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, a more accurate RTC time base timeouts (when LSI is used as the RTC clock source) and/or an IWDG timeout with an acceptable accuracy can be obtained.

### 7.5.10 General clock concept overview

The RCC handles the distribution of the CPU, bus interface and peripheral clocks for the system, according to the CPU operating mode (refer to **Section 7.5.1: Clock naming convention** for details on clock definitions).

For each peripheral, the application can control the activation/deactivation of its kernel and bus interface clock. Prior to use a peripheral, the CPU must enable it (by setting PERxEN to 1), and define if this peripheral remains active in Sleep mode (by setting PERxLPEN to 1). This is called ‘allocation’ of a peripheral by the CPU (refer to **Section 7.5.11: Peripheral allocation** for more details).

The peripheral allocation is used:
- by the RCC to automatically control the clock gating according to the CPU modes, and
- by the PWR to control the supply voltages of VCORE.

#### Memory handling

The CPU can access all the memory areas available in the product:

- AXISRAM1, AXISRAM2, AXISRAM3, AXISRAM4, ITCM, DTCM1, DTCM2 and FLASH
- AHBSRAM1 and AHBSRAM2
- BKPRAM

The BKPRAM, AHBSRAM1 and AHBSRAM2 have dedicated enable bits in order to gate the bus interface clock. The CPU needs to enable them prior to use these memories.

*Note:* The memory interface clocks (Flash memory and RAM interfaces) can be stopped by software during Sleep mode (via SRAMyLPEN bits).

Refer to **Section 7.5.12: Peripheral clock gating control** and **Section 7.5.13: CPU and bus matrix clock gating control** sections for details on clock enabling.
7.5.11 Peripheral allocation

The CPU can allocate a peripheral and hence control its kernel and bus interface clock.

The CPU can allocate a peripheral by setting the dedicated PERxEN bit to 1.

The CPU can control the peripheral clocks gating when it is in Sleep mode via the PERxLPEN bits.

Refer to for additional information.

The peripheral allocation bits (PERxEN bits) are used by the hardware to provide the kernel and bus interface clocks to the peripherals. However they are also used to link peripherals to the CPU. In this way, the hardware is able to safely gate the peripheral clocks and bus matrix clocks according to CPU states.

Clock switches and gating

- Clock switching delays

  The input selected by the clock switches can be changed dynamically without generating spurs or timing violation. For example, if PERxSEL (in Figure 74) goes from 0 to 1, the switch first disables the clock output using the currently selected clock (in0_ck), and enables again the clock output using the new selected clock (in1_ck). The disable and enable commands are re-synchronized to their respective clocks. If one of the two clocks are not present, the sequence cannot be completed, and no clock is output. To recover from this situation, the user must either provide a valid clock to in1_ck input or set back PERxSEL to 0.

  During the transition from one input to another, the kernel clock provided to the peripheral is gated, in the worst case, during 2 or 3 clock cycles of the new selected clock. As shown in Figure 74, both input clocks must be present during transition time.

- Clock enabling delays

  In the same way, the clock gating logic synchronizes the enable command (coming generally from a kernel clock request or PERxEN bits) with the selected clock, in order to avoid generation of spurs.

  - A maximum delay of two periods of the enabled clock may occur between the enable command and the first rising edge of the clock. The enable command can
be the rising edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request asserted by a peripheral.

– A maximum delay of 1.5 periods of the disabled clock may occur between the disable command and the last falling edge of the clock. The disable command can be the falling edge of the PERxEN bits of RCC_xxxxENR registers, or a kernel clock request released by a peripheral.

Note: Both the kernel and bus interface clocks are affected by this re-synchronization delay.

7.5.12 Peripheral clock gating control

As mentioned previously, each peripheral requires one or several bus interface clock, named rcc_perx_bus_ck (for peripheral ‘x’). These clocks can be an APB, AHB or AXI clock, according to which bus or busses, the peripheral is connected.

The clocks used as bus interface for peripherals, can be aclk, hclk[5:1], pclk[5:4], or pclk[2:1], depending on the bus connected to each peripheral.

Some peripherals also require dedicated clocks for their communication interface. These clocks are generally asynchronous with respect to the bus interface clock. They are named kernel clocks (perx_ker_ck). Both bus interface and kernel clocks can be gated according to several conditions detailed hereafter.

As shown in Figure 75, enabling the kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN and PERxLPEN bits,
  PERxEN represents the peripheral enable (allocation) bit for the CPU. The CPU can write these bits to 1 via the RCC_xxxxENR registers.
- CPU state (cpu_sleep and cpu_deepsleep signals)
- Kernel clock request (perx_ker_ckreq) of the peripheral itself, when the feature is available

Refer to Section 7.5.11: Peripheral allocation for more details.

Figure 75. Peripheral kernel clock enable logic details
Table 68 gives a detailed description of the enabling logic of the peripheral clocks for peripherals located in the CPU domain and allocated by the CPU.

Table 68. Peripheral clock enabling

<table>
<thead>
<tr>
<th>PERxEN</th>
<th>PERxLPEN</th>
<th>PERxSEL</th>
<th>perx_ker_ckreq</th>
<th>CPU state</th>
<th>rcc_perx_ker_ck</th>
<th>rcc_perx_bus_ck</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>No clock provided to the peripheral, because PERxEN=0</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Run</td>
<td>CK</td>
<td>CK</td>
<td>Kernel and bus interface clocks are provided to the peripheral, because the CPU is in Run, and PERxEN = 1.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Sleep</td>
<td>-</td>
<td>-</td>
<td>No clock provided to the peripheral, because the CPU is in Sleep and PERxLPEN = 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Sleep</td>
<td>CK</td>
<td>CK</td>
<td>Kernel and bus interface clocks are provided to the peripheral, because CPU is in Sleep, and PERxLPEN = 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Stop</td>
<td>-</td>
<td>-</td>
<td>No clock provided to the peripheral because the PERxLPEN bit is set to 0.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Stop</td>
<td>-</td>
<td>-</td>
<td>No clock provided to the peripheral because CPU is in Stop and lse_ck or lsi_ck or hsi_ker_ck or csi_ker_ck are not selected as kernel clock.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Stop</td>
<td>-</td>
<td>-</td>
<td>Kernel clock is provided to the peripheral because: PERxEN = PERxLPEN = 1 and lsi_ck or lse_ck are selected and enabled. The bus interface clock is not provided as the CPU is in Stop.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Stop</td>
<td>-</td>
<td>-</td>
<td>Kernel clock is provided to the peripheral because: req_ker_perx = 1, and PERxEN = PERxLPEN = 1 and hsi_ker_ck or csi_ker_ck are selected and enabled. The bus interface clock is not provided as the CPU is in Stop.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Stop</td>
<td>-</td>
<td>-</td>
<td>No clock provided to the peripheral because CPU is in Stop, and no kernel clock request pending</td>
</tr>
</tbody>
</table>

As a summary, we can state that the kernel clock is provided to the peripherals when the following conditions are met:
1. The CPU is in Run mode, and the peripheral is enabled.
2. The CPU is in Sleep mode and the peripheral is enabled with PERxLPEN = 1.
3. The CPU is in Stop mode, the peripheral is enabled with PERxLPEN = 1, the peripheral generates a kernel clock request and the selected clock is hsi_ker_ck or csi_ker_ck.
4. The CPU is in Stop mode, the peripheral is enabled with PERxLPEN = 1 and the kernel source clock of the peripheral is lse_ck or lsi_ck.

The bus interface clock is provided to the peripherals only when conditions 1 or 2 are met.
The RNG block is not clocked as other peripherals, the table hereafter shows the way RNG is clocked.

<table>
<thead>
<tr>
<th>RNGEN</th>
<th>RNGLPEN</th>
<th>rs48k_countries</th>
<th>CPU state</th>
<th>hsi48_ker_ck</th>
<th>rcc_rng_hclk3</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 X X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>No clock provided to the peripheral, because RNGEN=0</td>
</tr>
<tr>
<td>1 X 0</td>
<td></td>
<td>Run</td>
<td>- CK</td>
<td>hsi48_ker_ck</td>
<td>-</td>
<td>not provided because the kernel clock request is not asserted. hclk3 clock is provided</td>
</tr>
<tr>
<td>1 X 1</td>
<td></td>
<td>CK</td>
<td>CK</td>
<td>Both hsi48_ker_ck and hclk3 are provided because the kernel clock request is asserted, and RNGEN = 1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 X</td>
<td></td>
<td>Sleep</td>
<td>- CK</td>
<td>-</td>
<td>-</td>
<td>No clock provided, because the CPU is in Sleep and RNGLPEN = 0</td>
</tr>
<tr>
<td>1 1 0</td>
<td></td>
<td>Sleep</td>
<td>CK</td>
<td>hsi48_ker_ck</td>
<td>-</td>
<td>not provided because the kernel clock request is not asserted. hclk3 clock is provided</td>
</tr>
<tr>
<td>1 1 1</td>
<td></td>
<td></td>
<td>CK</td>
<td>Both hsi48_ker_ck and hclk3 are provided because the kernel clock request is asserted, RNGEN = 1 and RNGLPEN = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 X X</td>
<td></td>
<td>Stop</td>
<td>- CK</td>
<td>-</td>
<td>-</td>
<td>No clock provided because the CPU is in Stop mode</td>
</tr>
</tbody>
</table>
7.5.13  CPU and bus matrix clock gating control

The clocks of the CPU, AHB and AXI bridges and APB busses are enabled according to the rules hereafter:

- The CPU clock (rcc_cpu_ck) is enabled when the CPU is in Run or Sleep mode.
- The AXI bridge clock is enabled when the CPU is in Run or Sleep mode.
- Each AXI master and slave has an independent clock gating activated by default, to further reduce the power consumption. When activated the clock is automatically enabled on bus transaction request. The RCC AXI clocks gating disable register (RCC_CKGDISR) allows the application to control this behavior.
- The AHB bus matrix clock is enabled if one of the following conditions is met:
  - The CPU is in Run mode
  - The CPU is in Sleep mode, and at least one peripheral connected to this bridge have both its PERxEN and PERxLPEN set to 1.
  - The CPU is in Sleep mode, and AHB1, AHB2, AHB3, AHB4 or AHB5 has its clock enabled
- The clocks of the bridges AHB1, 2, 3, 4, 5 are enabled when one of the following conditions is met:
  - The CPU is in Run mode.
  - The CPU is in Sleep mode with at least one peripheral connected to this bus having both its PERxEN and PERxLPEN set to 1.
  - The CPU is in Sleep mode and the APB bus connected to the AHB bridge has its clock enabled.
- The APB1, 2, 4, 5 clock busses are enabled when one of the following conditions is met:
  - The CPU is in Run mode.
  - When the CPU is in Sleep mode with at least one peripheral connected to this bus having both its PERxEN and PERxLPEN set to 1.
7.5.14 Low-power emulation modes

In order to ease the debugging of the circuit, the RCC is able to handle an emulation mode for Stop and Standby modes.

Sleep emulation mode

The Sleep emulation mode is controlled by the DBG_SLEEP bit of the DBGMCU_CR register. When the processor goes to Sleep with DBG_SLEEP = 1, then the processor clock, the clocks of all enabled peripherals, debug parts, and interconnect are maintained activated.

Stop emulation mode

The Stop emulation mode is controlled by the DBG_STOP bit of DBGMCU_CR register. When the processor goes to Stop with DBG_STOP = 1, then:

- The PWR and the RCC remains activated.
- The VDDCORE voltage is set to the minimum VOS value.
- The RCC selects HSI or CSI as system clock, depending on STOPWUCK bit.
- The processor clock is gated (rcc_cpu_ck, see Figure 51),
- The free-running clock is generated (rcc_fclk, see Figure 51), and all the debug parts remain clocked.
- All peripheral clocks are gated. Peripherals can still receive a kernel clock if they request it, and if the kernel clock source is an RC oscillator. They also receive the kernel clock if the kernel clock source is LSI or LSE.
- The interconnect clocks are gated.

When a wakeup event occurs:

- The VDDCORE voltage remains to the minimum VOS value.
- The peripheral waking up the system sends the interrupt to the NVIC via the EXTI.
- The CPU exits from STOP and the RCC provide the processor clock and interconnect clocks.

Standby emulation mode

The Standby emulation mode is controlled by the DBG_STANDBY bit of the DBGMCU_CR register.

When the system goes to Standby with DBG_STANDBY = ‘1’, then:

- The VDDCORE voltage is not switched-off, but remains to the minimum VOS value.
- All the VDDCORE domain is under reset except the debug part.
- The debug part is clocked.
- Peripherals on VBat or VDD domains are not reset.

When the system exits from Standby, then:

- Option bytes are reloaded.
- The reset of the VDDCORE domain is released.
- The Standby flag is updated as well.
7.6 RCC interrupts

The RCC provides three interrupt lines:

- **rcc_it**: general interrupt line providing events when the PLLs are ready or when the oscillators are ready
- **rcc_hsecss_it**: interrupt line dedicated to the failure detection of the HSE CSS (clock security system)
- **rcc_lsecss_it**: interrupt line dedicated to the failure detection of the LSE CSS

The interrupt enable is controlled via **RCC clock source interrupt enable register** (*RCC_CIER*), except for the HSE CSS failure. When the HSE CSS feature is enabled, it is not possible to mask the interrupt generation.

The interrupt flags can be checked via **RCC clock source interrupt flag register** (*RCC_CIFR*), and these flags can be cleared via **RCC clock source interrupt clear register** (*RCC_CICR*).

**Note:** The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

**Table 70** gives a summary of the interrupt sources and the way to control them.

<table>
<thead>
<tr>
<th>Interrupt source</th>
<th>Description</th>
<th>Interrupt enable</th>
<th>Action to clear interrupt</th>
<th>Interrupt line</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSIRDYF</td>
<td>LSI ready</td>
<td>LSIRDYIE</td>
<td>Set LSIRDYC to 1</td>
<td>rcc_it</td>
</tr>
<tr>
<td>LSERDYF</td>
<td>LSE ready</td>
<td>LSERDYIE</td>
<td>Set LSERDYC to 1</td>
<td></td>
</tr>
<tr>
<td>HSIRDYF</td>
<td>HSI ready</td>
<td>HSIDRYIE</td>
<td>Set HSIRDYC to 1</td>
<td></td>
</tr>
<tr>
<td>HSERDYF</td>
<td>HSE ready</td>
<td>HSERDYIE</td>
<td>Set HSERDYC to 1</td>
<td></td>
</tr>
<tr>
<td>CSIRDYF</td>
<td>CSI ready</td>
<td>CSIRDYIE</td>
<td>Set CSIRDYC to 1</td>
<td></td>
</tr>
<tr>
<td>HSI48RDYF</td>
<td>HSI48 ready</td>
<td>HSI48RDYIE</td>
<td>Set HSI48RDYC to 1</td>
<td></td>
</tr>
<tr>
<td>PLL1RDYF</td>
<td>PLL1 ready</td>
<td>PLL1RDYIE</td>
<td>Set PLL1RDYC to 1</td>
<td></td>
</tr>
<tr>
<td>PLL2RDYF</td>
<td>PLL2 ready</td>
<td>PLL2RDYIE</td>
<td>Set PLL2RDYC to 1</td>
<td></td>
</tr>
<tr>
<td>PLL3RDYF</td>
<td>PLL3 ready</td>
<td>PLL3RDYIE</td>
<td>Set PLL3RDYC to 1</td>
<td></td>
</tr>
<tr>
<td>LSECSSF</td>
<td>LSE CSS failure</td>
<td>LSECSSFIE (1)</td>
<td>Set LSECSSC to 1</td>
<td>rcc_lsecss_it</td>
</tr>
<tr>
<td>HSECSSF</td>
<td>HSE CSS failure</td>
<td></td>
<td>Set HSECSSC to 1</td>
<td>rcc_hsecss_it</td>
</tr>
</tbody>
</table>

1. The security system feature must also be enabled (LSECSSON = 1), in order to generate interrupts.
2. It is not possible to mask this interrupt when the security system feature is enabled (HSECSSON = 1).
7.7 RCC Programming examples

7.7.1 PLL programming procedure

PLL initialization procedure

The recommended initialization sequence of the PLLs for the integer and fractional mode is given hereafter:

- The PLL is supposed to be disabled: PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN, PLLxTEN, PLLxON, and PLLxRDY set to '0', if it is not the case, refer to PLL disabling procedure.
- Enable the wanted clock source (HSE, HSI, or CSI) and wait for the ready flag.
- Select the clock source via PLLSCR into RCC PLLs clock source selection register (RCC_PLLCKSELR).
- Initialize the pre-dividers (DIVMx) in order to provide to the PLL, a valid reference clock. DIVMx are located into RCC PLLs clock source selection register (RCC_PLLCKSELR).
- Configure the PLL:
  - In integer or clock spreading mode the application must ensure that a 0 is loaded into the SDM. This can be done as follows:
    - Set PLLxFRACLE to 0
    - Write FRACN to 0
    - Set PLLxFRACLE to 1
    - Write FRACN to 0
    - Wait at least 5 µs
  - In fractional mode, the application must load the SDM with the wanted value (FracInitValue) by executing a sequence similar to the integer mode, but with FRACN = FracInitValue.
    - Program the fields PLLxRGE, PLLxVCOSEL, SSCG_CTRL, DIVxN, DIVxP, DIVxQ, DIVxR, DIV2S and DIV2T.
    - If the clock spreading mode is used, program also the fields MODPER, INCSTEP, SPREADSEL, TPDFNDIS, RPDFNDIS, and set PLLxSSCGEN to 1.
    - If the clock spreading mode is not used, set PLLxSSCGEN to 0.
  - Enable the PLL:
    - Set the corresponding PLLxON bit to 1, and wait for PLLxRDY bit set to 1.
    - If the PLL is in fractional mode, the FRACLE bit must not be set back to 0 as long as PLLxRDY = 0.
  - When the PLLxRDY bit is equal to 1, the application can enable the wanted outputs by setting PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN, PLLxTEN to 1.
  - If the application intends to tune the PLL frequency on-the-fly (only possible in fractional mode), then:
    - Set PLLxFRACLE to ‘0’,
      When FRACLE = ‘0’, the sigma delta modulator is still operating with the current value,
    - Write the new value into FRACN[12:0] (FracValue(n)),
    - Set PLLxFRACLE to 1, in order to latch the content of FRACN[12:0] into the SDM.
– Wait at least 5 µs.

**Note:** When the PLLxRDY goes to 1, it means that the PLLx output frequency is within 2% of its target value.

### Reprogramming post-dividers

When the PLLs are enabled, it is possible to change the values of a post-dividers (DIVP, DIVQ, DIVR, DIVS or DIVT) without disabling the corresponding PLLx, by performing the following sequence:

- Disable the wanted output by setting the bit PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN or PLLxTEN to 0,
- Change the value of the corresponding post-divider (DIVP, DIVQ, DIVR, DIVS or DIVT),
- Re-enable this output by setting the bit PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN or PLLxTEN to 1.

### PLL disabling procedure

The recommended disabling sequence is given hereafter:

- Disable all the post-dividers by setting PLLxPEN, PLLxQEN, PLLxREN, PLLxSEN and PLLxTEN bits of PLLx to 0,
- Disable the PLL: set PLLxON bit of PLLx to 0,
- Wait for PLLxRDY = 0.

### Using the SSCG block

For example, if an application needs to generate with the PLL2, a clock signal having the following characteristics:

- A frequency at DIVS output \((F_{pll2\_s\_ck})\) of 200 MHz Max,
- With a modulation depth of 0.5% peak \((M_D)\),
- With a frequency modulation of 28 kHz \((F_{MOD})\).

The assumption is that the crystal oscillator is 8 MHz \((F_{hse\_ck})\).

DIVM2 is programmed to 0 in order to provide a reference clock of 8 MHz to the PLL2 \((F_{ref2\_ck})\). The VCOH must be selected when SSCG is used.

From a 8 MHz reference clock, it is possible to generate 200 MHz, by programming the PLL2 as follow: \((DIVN+1) = 50\), \((F_{vco} = 800\text{ MHz})\), \((DIVS+1) = 4\).

The application has to compute the following parameters:

**Computing MODPER:**

\[
MODPER = \text{ROUND}\left(\frac{F_{ref\_ck}}{4 \times F_{MOD}}\right) = \text{ROUND}\left(\frac{8 \times 10^6}{4 \times 28 \times 10^3}\right) = 71
\]

**Computing INCSTEP:**

\[
\text{INCSTEP} = \text{ROUND}\left(\frac{(2^{15} - 1) \times M_D \times (DIVN + 1)}{100 \times 5 \times \text{MODPER}}\right) = \text{ROUND}\left(\frac{32767 \times 0.5 \times (49 + 1)}{100 \times 5 \times 71}\right) = 23
\]
Check that $\text{MODPER} \times \text{INCSTEP}$ is lower than $2^{15} - 1$.

In the example, $\text{MODPER} \times \text{INCSTEP} = 1633 \Rightarrow \text{OK}$

Due to rounding operations, the modulation period and the modulation depth does not completely match the target. The real modulation depth is:

$$M_D(\%) = \frac{\text{MODPER} \times \text{INCSTEP} \times 100 \times 5}{(2^{15} - 1) \times (\text{DIVN} + 1)} = \frac{71 \times 23 \times 100 \times 5}{(2^{15} - 1) \times (49 + 1)} = 0.498 \%$$

And the modulation frequency ($F_{\text{mod}}$) is:

$$F_{\text{mod}} = \frac{F_{\text{ck_ref}}}{4 \times \text{MODPER}} = \frac{8 \times 10^6}{4 \times 80} = 28.2 \text{ kHz}$$

The down-spread modulation must be used ($\text{SPREADSEL} = 1$) to keep the maximum frequency to a value lower than 200 MHz.

### 7.7.2 Frequency configuration examples

This section gives various frequency settings for CPU and XSPI interfaces, combined with audio constraints. The case without audio constraints is not shown as it is obvious that PLL1 and PLL2 can provide the expected frequency. If audio applications are required, several options are offered to the user:

- Generate the kernel clock with PLL1, PLL2 or PLL3
- Use the external audio clock input (I2S_CKIN)
- Use oscillators

In this section, we explore the cases where PLL1 or PLL2 are used for audio.

Note as well that PLL2 is also used for XSPIs, if it is shared for audio application, it is not recommended to activate the SSCG.

The configurations listed in the tables hereafter are not exhaustive, many other setting are available. For example if the clock provided to the SAI must be shared with FDCAN, then the application would like to generate a clock around 100 MHz to insure good working conditions for FDCAN. These considerations are not taken into account in these tables.

Note also that in order to get the best clock quality at PLL output, $F_{\text{REF}}$ must be as close as possible to 16 MHz. This is not always easy to do for PLLs in integer mode, but can be done easily when the PLLs are in fractional mode (as shown in the tables).
### Table 71. Clock configuration examples with PLLs in integer mode

<table>
<thead>
<tr>
<th>Configuration</th>
<th>HSE (MHz)</th>
<th>DIVM+1</th>
<th>FREF (MHz)</th>
<th>PLLXRGGE</th>
<th>DIVN+1</th>
<th>FRACV</th>
<th>FVCO (MHz)</th>
<th>DIVP+1</th>
<th>DIVQ+1</th>
<th>DIVS/T+1</th>
<th>FCPU (MHz)</th>
<th>FXSPI (MHz)</th>
<th>FAudio (MHz)</th>
<th>FCPU accuracy (%)</th>
<th>FXSPI accuracy (%)</th>
<th>FAudio accuracy (ppm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL1</td>
<td>CPU: 500 MHz</td>
<td>16</td>
<td>7</td>
<td>2.286</td>
<td>1</td>
<td>215</td>
<td>0</td>
<td>982.9</td>
<td>1</td>
<td>4</td>
<td>491.4</td>
<td>-</td>
<td>122.86</td>
<td>-1.7</td>
<td>-</td>
<td>-186</td>
</tr>
<tr>
<td></td>
<td>Audio: 48 kHz</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PLL2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>XSPI: 200 MHz</td>
<td>16</td>
<td>7</td>
<td>2.286</td>
<td>1</td>
<td>258</td>
<td>0</td>
<td>1179.4</td>
<td>1</td>
<td>6</td>
<td>589.7</td>
<td>-</td>
<td>98.290</td>
<td>-1.7</td>
<td>-</td>
<td>-186</td>
</tr>
<tr>
<td></td>
<td>XSPI: 166 MHz</td>
<td>16</td>
<td>7</td>
<td>2.286</td>
<td>1</td>
<td>215</td>
<td>0</td>
<td>962.9</td>
<td>4</td>
<td>3</td>
<td>163.8</td>
<td>-</td>
<td>122.86</td>
<td>-1.7</td>
<td>-</td>
<td>-186</td>
</tr>
<tr>
<td></td>
<td>XSPI: 166 MHz</td>
<td>16</td>
<td>8</td>
<td>2.000</td>
<td>1</td>
<td>333</td>
<td>0</td>
<td>1332.0</td>
<td>59</td>
<td>-</td>
<td>166.5</td>
<td>-</td>
<td>11.288</td>
<td>-0.1</td>
<td>-</td>
<td>-130</td>
</tr>
</tbody>
</table>

1. FREF is the reference frequency at PLL1 or PLL2 inputs
2. FVCO is the clock frequency at VCO output of PLL1 or PLL2
3. F_CPU is the clock frequency provided to the CPU (rcc_cpu_ck)
4. FXSPI is the kernel clock frequency provided to the XSPI1 or XSPI2 (xspi_ker_ck)
5. F_AUDIO is the kernel clock frequency provided to the audio blocks SAIx, SPI/I2Sx,...

For a 48 kHz stream, the expected kernel clock is 48 kHz x 256 x k, with k integer
For a 44.1 kHz stream, the expected kernel clock is 44.1 kHz x 256 x k, with k integer
### Table 72. Clock configuration examples with PLLs in fractional mode\(^{(1)}\)

<table>
<thead>
<tr>
<th>Configuration</th>
<th>HSE (MHz)</th>
<th>DIVM+1</th>
<th>DIVN+1</th>
<th>FRACV</th>
<th>F_VCO (MHz) (^{(2)})</th>
<th>DIVP+1</th>
<th>DIVQ+1</th>
<th>DIVS/T+1</th>
<th>F_CPU (MHz)</th>
<th>F_XSPI (MHz) (^{(2)})</th>
<th>F_AUDIO (MHz) (^{(6)})</th>
<th>F_CPU accuracy (%)</th>
<th>F_XSPI accuracy (%)</th>
<th>F_AUDIO accuracy (ppm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLL1 CPU: 500 MHz Audio: 48 kHz</td>
<td>16 1 16.0 30 5898</td>
<td>983.0</td>
<td>1 4</td>
<td>-</td>
<td>491.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>122.88</td>
<td>-1.7</td>
<td>-1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PLL1 CPU: 500 MHz Audio: 44.1 kHz</td>
<td>16 1 16.0 31 380</td>
<td>993.5</td>
<td>1 4</td>
<td>-</td>
<td>496.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>124.2</td>
<td>-0.6</td>
<td>-1.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PLL1 CPU: 550 MHz Audio: 48 kHz</td>
<td>16 1 16.0 33 6488</td>
<td>1081.3</td>
<td>1 11</td>
<td>-</td>
<td>540.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>49.15</td>
<td>-1.7</td>
<td>-</td>
<td>0.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PLL1 CPU: 550 MHz Audio: 44.1 kHz</td>
<td>16 1 16.0 33 7117</td>
<td>1083.8</td>
<td>1 6</td>
<td>-</td>
<td>541.9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>90.32</td>
<td>-1.5</td>
<td>-</td>
<td>1.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PLL1 CPU: 600 MHz Audio: 48 kHz</td>
<td>16 1 16.0 36 7078</td>
<td>1179.6</td>
<td>1 6</td>
<td>-</td>
<td>589.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>98.30</td>
<td>-1.7</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PLL1 CPU: 600 MHz Audio: 44.1 kHz</td>
<td>16 1 16.0 36 5662</td>
<td>1174.1</td>
<td>1 13</td>
<td>-</td>
<td>587.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>45.15</td>
<td>-2.1</td>
<td>-</td>
<td>-1.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PLL2 XSPI: 200 MHz Audio: 48 kHz</td>
<td>16 1 16.0 36 7078</td>
<td>1179.7</td>
<td>6</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>196.6</td>
<td>98.304</td>
<td>-</td>
<td>-1.7</td>
<td>-</td>
<td>-0.5</td>
</tr>
<tr>
<td>PLL2 XSPI: 200 MHz Audio: 44.1 kHz</td>
<td>16 1 16.0 36 5662</td>
<td>1174.1</td>
<td>13</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>195.7</td>
<td>45.158</td>
<td>-</td>
<td>-2.1</td>
<td>-</td>
<td>-1</td>
</tr>
<tr>
<td>PLL2 XSPI: 166 MHz Audio: 48 kHz</td>
<td>16 1 16.0 30 5898</td>
<td>983.0</td>
<td>4</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>163.8</td>
<td>122.88</td>
<td>-</td>
<td>-1.7</td>
<td>-</td>
<td>-0.9</td>
</tr>
<tr>
<td>PLL2 XSPI: 166 MHz Audio: 44.1 kHz</td>
<td>16 1 16.0 31 380</td>
<td>993.5</td>
<td>4</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>165.6</td>
<td>124.186</td>
<td>-</td>
<td>-0.7</td>
<td>-</td>
<td>-1.4</td>
</tr>
</tbody>
</table>

1. Note that for all these configurations, PLLxRGE is always equal to 3.
2. \(F_{\text{REF}}\) is the reference frequency at PLL1 or PLL2 inputs.
3. \(F_{\text{VCO}}\) is the clock frequency at VCO output of PLL1 or PLL2
4. \(F_{\text{CPU}}\) is the clock frequency provided to the CPU (rcc_cpu_ck)
5. \(F_{\text{XSPI}}\) is the kernel clock frequency provided to the XSPI1 or XSPI2 (xspi_ker_ck)
6. \(F_{\text{AUDIO}}\) is the kernel clock frequency provided to the audio blocks SAIx, SPI/I2Sx...
   For a 48 kHz stream, the expected kernel clock is 48 kHz x 256 x k, with k integer.
   For a 44.1 kHz stream, the expected kernel clock is 44.1 kHz x 256 x k, with k integer.
## 7.8 RCC registers

### 7.8.1 RCC source control register (RCC_CR)

Address offset: 0x000  
Reset value: 0x0000 0025

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Bit 31:30 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bit 29 PLL3RDY: PLL3 clock ready flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware to indicate that the PLL3 is locked.</td>
</tr>
<tr>
<td></td>
<td>0: PLL3 unlocked (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: PLL3 locked</td>
</tr>
<tr>
<td></td>
<td>Bit 28 PLL3ON: PLL3 enable</td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software to enable PLL3.</td>
</tr>
<tr>
<td></td>
<td>Cleared by hardware when entering Stop or Standby mode.</td>
</tr>
<tr>
<td></td>
<td>0: PLL3 OFF (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: PLL3 ON</td>
</tr>
<tr>
<td></td>
<td>Bit 27 PLL2RDY: PLL2 clock ready flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware to indicate that the PLL2 is locked.</td>
</tr>
<tr>
<td></td>
<td>0: PLL2 unlocked (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: PLL2 locked</td>
</tr>
<tr>
<td></td>
<td>Bit 26 PLL2ON: PLL2 enable</td>
</tr>
<tr>
<td></td>
<td>Set and cleared by software to enable PLL2.</td>
</tr>
<tr>
<td></td>
<td>Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if FMCCKP = 1, or XSPICKP = 1.</td>
</tr>
<tr>
<td></td>
<td>0: PLL2 OFF (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: PLL2 ON</td>
</tr>
<tr>
<td></td>
<td>Bit 25 PLL1RDY: PLL1 clock ready flag</td>
</tr>
<tr>
<td></td>
<td>Set by hardware to indicate that the PLL1 is locked.</td>
</tr>
<tr>
<td></td>
<td>0: PLL1 unlocked (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: PLL1 locked</td>
</tr>
</tbody>
</table>
Bit 24 **PLL1ON**: PLL1 enable
Set and cleared by software to enable PLL1.
Cleared by hardware when entering Stop or Standby mode. Note that the hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock (SW=3) or if FMCKP = 1, or if XSPICKP = 1.
0: PLL1 OFF (default after reset)
1: PLL1 ON

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **HSECSSON**: HSE clock security system enable
Set by software to enable clock security system on HSE.
This bit is "set only" (disabled by a system reset or when the system enters in Standby mode).
When HSECSSON is set, the clock detector is enabled by hardware when the HSE is ready and disabled by hardware if an oscillator failure is detected.
0: CSS on HSE OFF (clock detector OFF) (default after reset)
1: CSS on HSE ON (clock detector ON if the HSE oscillator is stable, OFF if not).

Bit 19 **HSEEXT**: external high speed clock type in Bypass mode
Set and reset by software to select the external clock type (analog or digital).
The external clock must be enabled with the HSEON bit to be used by the device.
The HSEEXT bit can be written only if the HSE oscillator is disabled.
0: HSE in analog mode (default after reset)
1: HSE in digital mode

Bit 18 **HSEBYP**: HSE clock bypass
Set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSEON bit to be used by the device.
The HSEBYP bit can be written only if the HSE oscillator is disabled.
0: HSE oscillator not bypassed (default after reset)
1: HSE oscillator bypassed with an external clock

Bit 17 **HSERDY**: HSE clock ready flag
Set by hardware to indicate that the HSE oscillator is stable.
0: HSE clock is not ready (default after reset)
1: HSE clock is ready

Bit 16 **HSEON**: HSE clock enable
Set and cleared by software.
Cleared by hardware to stop the HSE when entering Stop or Standby mode.
This bit cannot be cleared if the HSE is used directly (via SW mux) as system clock, or if the HSE is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1) or if FMCKP = 1, or if XSPICKP = 1.
0: HSE is OFF (default after reset)
1: HSE is ON

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

Bit 13 **HSI48RDY**: HSI48 clock ready flag
Set by hardware to indicate that the HSI48 oscillator is stable.
0: HSI48 clock is not ready (default after reset)
1: HSI48 clock is ready
Bit 12  **HSI48ON**: HSI48 clock enable
    Set by software and cleared by software or by the hardware when the system enters to Stop or Standby mode.
    0: HSI48 is OFF (default after reset)
    1: HSI48 is ON

Bits 11:10 Reserved, must be kept at reset value.

Bit 9  **CSIKERON**: CSI clock enable in Stop mode
    Set and reset by software to force the CSI to ON, even in Stop mode, in order to be quickly available as kernel clock for some peripherals. This bit has no effect on the value of CSION.
    0: no effect on CSI (default after reset)
    1: CSI is forced to ON even in Stop mode

Bit 8  **CSIRDY**: CSI clock ready flag
    Set by hardware to indicate that the CSI oscillator is stable. This bit is activated only if the RC is enabled by CSION (it is not activated if the CSI is enabled by CSIKERON or by a peripheral request).
    0: CSI clock is not ready (default after reset)
    1: CSI clock is ready

Bit 7  **CSION**: CSI clock enable
    Set and reset by software to enable/disable CSI clock for system and/or peripheral.
    Set by hardware to force the CSI to ON when the system leaves Stop mode, if STOPWUCK = 1 or STOPKERWUCK = 1.
    This bit cannot be cleared if the CSI is used directly (via SW mux) as system clock, or if the CSI is selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1) or if FMCCKP = 1, or if XSPICKP = 1.
    0: CSI is OFF (default after reset)
    1: CSI is ON

Bit 6  Reserved, must be kept at reset value.

Bit 5  **HSIDIVF**: HSI divider flag
    Set and reset by hardware.
    As a write operation to HSIDIV has not an immediate effect on the frequency, this flag indicates the current status of the HSI divider. HSIDIVF goes immediately to 0 when HSIDIV value is changed, and is set back to 1 when the output frequency matches the value programmed into HSIDIV.
    0: new division ratio not yet propagated to *hsi(_ker)_ck* (default after reset)
    1: *hsi(_ker)_ck* clock frequency reflects the new HSIDIV value (default register value when the clock setting is completed)

Bits 4:3  **HSIDIV[1:0]**: HSI clock divider
    Set and reset by software.
    These bits allow selecting a division ratio in order to configure the wanted HSI clock frequency. The HSI48 cannot be changed if the HSI is selected as reference clock for at least one enabled PLL (PLLxON bit set to 1). In that case, the new HSIDIV value is ignored.
    00: division by 1, *hsi(_ker)_ck* = 64 MHz (default after reset)
    01: division by 2, *hsi(_ker)_ck* = 32 MHz
    10: division by 4, *hsi(_ker)_ck* = 16 MHz
    11: division by 8, *hsi(_ker)_ck* = 8 MHz
Bit 2  HSIRDY: HSI clock ready flag
    Set by hardware to indicate that the HSI oscillator is stable.
    0: HSI clock is not ready (default after reset)
    1: HSI clock is ready

Bit 1  HSIKERON: HSI clock enable in Stop mode
    Set and reset by software to force the HSI to ON, even in Stop mode, in order to be quickly available
    as kernel clock for peripherals. This bit has no effect on the value of HSION.
    0: no effect on HSI (default after reset)
    1: HSI is forced to ON even in Stop mode

Bit 0  HSION: HSI clock enable
    Set and cleared by software.
    Set by hardware to force the HSI to ON when the product leaves Stop mode, if STOPWUCK = 0 or
    STOPKERWUCK = 0.
    Set by hardware to force the HSI to ON when the product leaves Standby mode or in case of a
    failure of the HSE which is used as the system clock source.
    This bit cannot be cleared if the HSI is used directly (via SW switch) as system clock, or if the HSI is
    selected as reference clock for PLL1 with PLL1 enabled (PLL1ON bit set to 1) or if FMCCKP = 1, or
    if XSPICKP = 1.
    0: HSI is OFF
    1: HSI is ON (default after reset)
### 7.8.2 RCC clock protection register (RCC_CKPROTR)

Address offset: 0x100  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Bits 31:15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Bits 14:12</td>
<td>FMCSWP[2:0]: FMC kernel clock switch position</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>Set by hardware.</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>This field can be used to verify the real position of FMC kernel switch selector.</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>000: The switch is in neutral mode and output clock is gated (default after reset)</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>001: The switch is selecting hclk5</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>010: The switch is selecting pll1_q_ck</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>011: The switch is selecting pll2_r_ck</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>100: The switch is selecting hsi_ker_ck</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>101: The switch is in recovery position (hclk5/4)</td>
</tr>
<tr>
<td>21</td>
<td>Bit 11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Bits 10:8</td>
<td>XSPI2SWP[2:0]: XSPI2 kernel clock switch position</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>Set by hardware.</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>This field can be used to verify the real position of XSPI2 kernel switch selector.</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>000: The switch is in neutral mode and output clock is gated (default after reset)</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>001: The switch is selecting hclk5</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>010: The switch is selecting pll2_s_ck</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>011: The switch is selecting pll2_t_ck</td>
</tr>
<tr>
<td>13</td>
<td>Bit 7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Bits 6:4</td>
<td>XSPI1SWP[2:0]: XSPI1 kernel clock switch position</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>Set by hardware.</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>This field can be used to verify the real position of XSPI1 kernel switch selector.</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>000: The switch is in neutral mode and output clock is gated (default after reset)</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>001: The switch is selecting hclk5</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>010: The switch is selecting pll2_s_ck</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>011: The switch is selecting pll2_t_ck</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>100: The switch is in recovery position (hclk5/4)</td>
</tr>
</tbody>
</table>
7.8.3 RCC HSI calibration register (RCC_HSICFGR)

Address offset: 0x004

Reset value: 0x4000 0XXX

Reset value depends on the flash option bytes setting.

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>HSITRIM[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HSICAL[11:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **HSITRIM[6:0]**: HSI clock trimming

Set by software to adjust calibration.

HSITRIM field is added to the engineering option bytes loaded during reset phase (FLASH_HSI_opt) in order to form the calibration trimming value.

HSICAL = HSITRIM + FLASH_HSI_opt.

**Note:** The reset value of the field is 0x40.

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **HSICAL[11:0]**: HSI clock calibration

Set by hardware by option byte loading.

Adjusted by software through trimming bits HSITRIM.

This field represents the sum of engineering option byte calibration value and HSITRIM bits value.
### 7.8.4 RCC clock recovery RC register (RCC_CRRCCR)

Address offset: 0x008  
Reset value: 0x0000 0XXX  
Reset value depends on the flash option bytes setting.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

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

Bits 9:0  HSIC48CAL[9:0]: Internal RC 48 MHz clock calibration  
Set by hardware by option byte loading.  
Read-only.

### 7.8.5 RCC CSI calibration register (RCC_CSICFGR)

Address offset: 0x00C  
Reset value: 0x2000 0XXX  
Reset value depends on the flash option bytes setting.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bits 29:24  CSITRIM[5:0]: CSI clock trimming  
Set by software to adjust calibration.  
CSITRIM field is added to the engineering option bytes loaded during reset phase  
(FLASH_CSI_opt) in order to form the calibration trimming value.  
CSICAL = CSITRIM + FLASH_CSI_opt.  
*Note:* The reset value of the field is 0x20.

Bits 23:8  Reserved, must be kept at reset value.

Bits 7:0  CSICAL[7:0]: CSI clock calibration  
Set by hardware by option byte loading.  
Adjusted by software through trimming bits CSITRIM.  
This field represents the sum of engineering option byte calibration value and CSITRIM bits value.
7.8.6 RCC clock configuration register (RCC_CFGR)

Address offset: 0x010
Reset value: 0x0000 0000

Bits 31:29 **MCO2SEL[2:0]**: microcontroller clock output 2
Set and cleared by software. Clock source selection may generate glitches on MCO2.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
- 000: system clock selected (**sys_ck**) (default after reset)
- 001: PLL2 oscillator clock selected (**pll2_p_ck**)
- 010: HSE clock selected (**hse_ck**)
- 011: PLL1 clock selected (**pll1_p_ck**)
- 100: CSI clock selected (**csi_ck**)
- 101: LSI clock selected (**lsi_ck**)
- others: reserved

Bits 28:25 **MCO2PRE[3:0]**: MCO2 prescaler
Set and cleared by software to configure the prescaler of the MCO2. Modification of this prescaler may generate glitches on MCO2. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
- 0000: prescaler disabled (default after reset)
- 0001: division by 1 (bypass)
- 0010: division by 2
- 0011: division by 3
- 0100: division by 4
- ...
- 1111: division by 15

Bits 24:22 **MCO1SEL[2:0]**: Microcontroller clock output 1
Set and cleared by software. Clock source selection may generate glitches on MCO1.
It is highly recommended to configure these bits only after reset, before enabling the external oscillators and the PLLs.
- 000: HSI clock selected (**hsi_ck**) (default after reset)
- 001: LSE oscillator clock selected (**lse_ck**)
- 010: HSE clock selected (**hse_ck**)
- 011: PLL1 clock selected (**pll1_q_ck**)
- 100: HSI48 clock selected (**hsi48_ck**)
- others: reserved
Bits 21:18  **MCO1PRE[3:0]**: MCO1 prescaler
   Set and cleared by software to configure the prescaler of the MCO1. Modification of this prescaler may generate glitches on MCO1. It is highly recommended to change this prescaler only after reset, before enabling the external oscillators and the PLLs.
   0000: prescaler disabled (default after reset)
   0001: division by 1 (bypass)
   0010: division by 2
   0011: division by 3
   0100: division by 4
   ...
   1111: division by 15

Bits 17:16  Reserved, must be kept at reset value.

Bit 15  **TIMPRE**: timers clocks prescaler selection
   This bit is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.
   0: The timers kernel clock is equal to \( \text{rcc}_\text{hclk1} \) if \( \text{CDPPREx} \) is corresponding to division by 1 or 2, else it is equal to \( 2 \times \text{rcc}_\text{pclkx}_\text{d2} \) (default after reset)
   1: The timers kernel clock is equal to \( \text{rcc}_\text{hclk1} \) if \( \text{CDPPREx} \) is corresponding to division by 1, 2 or 4, else it is equal to \( 4 \times \text{rcc}_\text{pclkx}_\text{d2} \)
   Refer to  Table 67: Ratio between clock timer and pclk for more details.

Bit 14  Reserved, must be kept at reset value.

Bits 13:8  **RTCPRE[5:0]**: HSE division factor for RTC clock
   Set and cleared by software to divide the HSE to generate a clock for RTC.
   Caution: The software must set these bits correctly to ensure that the clock supplied to the RTC is lower than 1 MHz. These bits must be configured if needed before selecting the RTC clock source.
   000000: no clock (default after reset)
   000001: no clock
   000010: HSE/2
   000011: HSE/3
   000100: HSE/4
   ...
   111110: HSE/62
   111111: HSE/63

Bit 7  **STOPKERWUCK**: kernel clock selection after a wake up from system Stop
   Set and reset by software to select the kernel wakeup clock from system Stop.
   0: HSI selected as wake up clock from system Stop (default after reset)
   1: CSI selected as wake up clock from system Stop
   See  Section 1.: Dividers values can be changed on-the-fly. All dividers provide have 50% duty-cycles. for details.
Bit 6 STOPWUCK: system clock selection after a wake up from system Stop
  Set and reset by software to select the system wakeup clock from system Stop.
The selected clock is also used as emergency clock for the clock security system (CSS) on HSE.
  0: HSI selected as wake up clock from system Stop (default after reset)
  1: CSI selected as wake up clock from system Stop
  See Section 1.: Dividers values can be changed on-the-fly. All dividers provide have 50% duty-cycles. for details.
  Caution: STOPWUCK must not be modified when CSS is enabled (by HSECSSON bit) and the system clock is HSE (SWS = 10) or a switch on HSE is requested (SW =10).

Bits 5:3 SWS[2:0]: system clock switch status
  Set and reset by hardware to indicate which clock source is used as system clock.
  000: HSI used as system clock (hsi_ck) (default after reset)
  001: CSI used as system clock (csi_ck)
  010: HSE used as system clock (hse_ck)
  011: PLL1 used as system clock (pll1_p_ck)
  others: reserved

Bits 2:0 SW[2:0]: system clock switch
  Set and reset by software to select system clock source (sys_ck).
  Set by hardware in order to force the selection of the HSI or CSI (depending on STOPWUCK selection) when leaving a system Stop mode or in case of failure of the HSE when used directly or indirectly as system clock.
  000: HSI selected as system clock (hsi_ck) (default after reset)
  001: CSI selected as system clock (csi_ck)
  010: HSE selected as system clock (hse_ck)
  011: PLL1 selected as system clock (pll1_p_ck)
  others: reserved
7.8.7 RCC CPU domain clock configuration register (RCC_CDCFGR)

Address offset: 0x018
Reset value: 0x0000 0000

Caution:
Care must be taken when using the voltage scaling. Due to the propagation delay of the new division factor, after a prescaler factor change and before lowering the VCORE voltage, this register must be read in order to check that the new prescaler value has been taken into account.

Depending on the clock source frequency and the voltage range, the software application must program a correct value in BMPRE to make sure that the system frequency does not exceed the maximum frequency.
### 7.8.8 RCC AHB clock configuration register (RCC_BMCFGR)

Address offset: 0x01C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

BMPRE[3:0]

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

**Bits 3:0 BMPRE[3:0]: Bus matrix clock prescaler**

Set and reset by software to control the division factor of `rcc_hclk[5:1]` and `rcc_aclk`. This group of clocks is also named `sys_bus_ck`. Changing this division ratio has an impact on the frequency of all bus matrix clocks.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xxx</td>
<td><code>sys_bus_ck = sys_cpu_ck</code> (default after reset)</td>
</tr>
<tr>
<td>1000</td>
<td><code>sys_bus_ck = sys_cpu_ck / 2</code></td>
</tr>
<tr>
<td>1001</td>
<td><code>sys_bus_ck = sys_cpu_ck / 4</code></td>
</tr>
<tr>
<td>1010</td>
<td><code>sys_bus_ck = sys_cpu_ck / 8</code></td>
</tr>
<tr>
<td>1011</td>
<td><code>sys_bus_ck = sys_cpu_ck / 16</code></td>
</tr>
<tr>
<td>1100</td>
<td><code>sys_bus_ck = sys_cpu_ck / 64</code></td>
</tr>
<tr>
<td>1101</td>
<td><code>sys_bus_ck = sys_cpu_ck / 128</code></td>
</tr>
<tr>
<td>1110</td>
<td><code>sys_bus_ck = sys_cpu_ck / 256</code></td>
</tr>
<tr>
<td>1111</td>
<td><code>sys_bus_ck = sys_cpu_ck / 512</code></td>
</tr>
</tbody>
</table>

*Note:* The clocks are divided by the new prescaler factor from 1 to 16 periods of the slowest APB clock among `rcc_pclk1,2,4,5` after BMPRE update.

*Note:* Note also that frequency of `rcc_hclk[5:1] = rcc_aclk = sys_bus_ck`. 
7.8.9 RCC APB clocks configuration register (RCC_APBCFGR)

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE5[2:0]**: CPU domain APB5 prescaler
- Set and reset by software to control the division factor of rcc_pclk5.
- The clock is divided by the new prescaler factor from 1 to 16 cycles of sys_bus_ck after PPRE5 write.
  - 0xx: rcc_pclk5 = sys_bus_ck (default after reset)
  - 100: rcc_pclk5 = sys_bus_ck / 2
  - 101: rcc_pclk5 = sys_bus_ck / 4
  - 110: rcc_pclk5 = sys_bus_ck / 8
  - 111: rcc_pclk5 = sys_bus_ck / 16

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **PPRE4[2:0]**: CPU domain APB4 prescaler
- Set and reset by software to control the division factor of rcc_pclk4.
- The clock is divided by the new prescaler factor from 1 to 16 cycles of sys_bus_ck after PPRE4 write.
  - 0xx: rcc_pclk4 = sys_bus_ck (default after reset)
  - 100: rcc_pclk4 = sys_bus_ck / 2
  - 101: rcc_pclk4 = sys_bus_ck / 4
  - 110: rcc_pclk4 = sys_bus_ck / 8
  - 111: rcc_pclk4 = sys_bus_ck / 16

Bit 7 Reserved, must be kept at reset value.
Bits 6:4  **PPRE2[2:0]**: CPU domain APB2 prescaler
   Set and reset by software to control the division factor of **rcc_pclk2**.
   The clock is divided by the new prescaler factor from 1 to 16 cycles of **sys_bus_ck** after PPRE2 write.
   
   0xx: `rcc_pclk2 = sys_bus_ck` (default after reset)
   100: `rcc_pclk2 = sys_bus_ck / 2`
   101: `rcc_pclk2 = sys_bus_ck / 4`
   110: `rcc_pclk2 = sys_bus_ck / 8`
   111: `rcc_pclk2 = sys_bus_ck / 16`

   Bit 3  Reserved, must be kept at reset value.

Bits 2:0  **PPRE1[2:0]**: CPU domain APB1 prescaler
   Set and reset by software to control the division factor of **rcc_pclk1**.
   The clock is divided by the new prescaler factor from 1 to 16 cycles of **sys_bus_ck** after PPRE1 write.
   
   0xx: `rcc_pclk1 = sys_bus_ck` (default after reset)
   100: `rcc_pclk1 = sys_bus_ck / 2`
   101: `rcc_pclk1 = sys_bus_ck / 4`
   110: `rcc_pclk1 = sys_bus_ck / 8`
   111: `rcc_pclk1 = sys_bus_ck / 16`
7.8.10 RCC PLLs clock source selection register (RCC_PLLCKSELR)

Address offset: 0x028
Reset value: 0x0202 0200

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

Bits 25:20 **DIVM3[5:0]**: prescaler for PLL3
Set and cleared by software to configure the prescaler of the PLL3.
The hardware does not allow any modification of this prescaler when PLL3 is enabled (PLL3ON = 1).
In order to save power when PLL3 is not used, the value of DIVM3 must be set to 0.
000000: prescaler disabled
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
... 100000: division by 32 (default after reset)
... 111111: division by 63

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:12 **DIVM2[5:0]**: prescaler for PLL2
Set and cleared by software to configure the prescaler of the PLL2.
The hardware does not allow any modification of this prescaler when PLL2 is enabled (PLL2ON = 1).
In order to save power when PLL2 is not used, the value of DIVM2 must be set to 0.
000000: prescaler disabled
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
... 100000: division by 32 (default after reset)
... 111111: division by 63

Bits 11:10 Reserved, must be kept at reset value.
Bits 9:4 **DIVM1[5:0]**: prescaler for PLL1
Set and cleared by software to configure the prescaler of the PLL1.
The hardware does not allow any modification of this prescaler when PLL1 is enabled (PLL1ON = 1).
In order to save power when PLL1 is not used, the value of DIVM1 must be set to 0.
000000: prescaler disabled
000001: division by 1 (bypass)
000010: division by 2
000011: division by 3
...
100000: division by 32 (default after reset)
...
111111: division by 63

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **PLLSRC[1:0]**: DIVMx and PLLs clock source selection
Set and reset by software to select the PLL clock source.
These bits can be written only when all PLLs are disabled.
In order to save power, when no PLL is used, PLLSRC must be set to ‘11’.
00: HSI selected as PLL clock (**hsi_ck**)(default after reset)
01: CSI selected as PLL clock (**csi_ck**)
10: HSE selected as PLL clock (**hse_ck**)
11: no clock send to DIVMx divider and PLLs
### 7.8.11 RCC PLLs configuration register (RCC_PLLCFGR)

Address offset: 0x02C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26:25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>PLL3SEN</td>
<td>PLL3REN</td>
<td>PLL3QEN</td>
<td>PLL3[3:0]</td>
<td>PLL3SSCGEN</td>
<td>PLL3VCOSEL</td>
<td>PLL3FRACEN</td>
<td>Reserved</td>
<td>PLL2TEN</td>
<td>PLL2SEN</td>
<td>PLL2REN</td>
<td>PLL2QEN</td>
<td>PLL2[3:0]</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>PLL2[3:0]</td>
<td>PLL2SSCGEN</td>
<td>PLL2VCOSEL</td>
<td>PLL2FRACEN</td>
<td>Reserved</td>
<td>PLL2[3:0]</td>
<td>PLL2QEN</td>
<td>PLL2[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31** Reserved, must be kept at reset value.

**Bit 30 PLL3SEN:** PLL3 DIVS divider output enable  
Set and reset by software to enable the PLL3.  
To save power, PLL3SEN must be set to 0 when the PLL3 is not used.  
0: PLL3 output disabled (default after reset)  
1: PLL3 output enabled

**Bit 29 PLL3REN:** PLL3 DIVR divider output enable  
Set and reset by software to enable the PLL3.  
To save power, PLL3REN and DIVR bits must be set to 0 when the PLL3 is not used.  
0: PLL3 output disabled (default after reset)  
1: PLL3 output enabled

**Bit 28 PLL3QEN:** PLL3 DIVQ divider output enable  
Set and reset by software to enable the PLL3.  
To save power, PLL3REN and DIVQ bits must be set to 0 when the PLL3 is not used.  
0: PLL3 output disabled (default after reset)  
1: PLL3 output enabled

**Bit 27 PLL3PEN:** PLL3 DIVP divider output enable  
Set and reset by software to enable the PLL3.  
To save power, PLL3REN and DIVP bits must be set to 0 when the PLL3 is not used.  
0: PLL3 output disabled (default after reset)  
1: PLL3 output enabled

**Bits 26:25 PLL3[3:0]:** PLL3 input frequency range  
Set and reset by software to select the proper reference frequency range used for PLL3.  
These bits must be written before enabling the PLL3.  
00: PLL3 input (ref3_ck) clock range frequency between 1 and 2 MHz (default after reset)  
01: PLL3 input (ref3_ck) clock range frequency between 2 and 4 MHz  
10: PLL3 input (ref3_ck) clock range frequency between 4 and 8 MHz  
11: PLL3 input (ref3_ck) clock range frequency between 8 and 16 MHz
Bit 24 **PLL3SSCGEN**: PLL3 SSCG enable
Set and reset by software to enable the Spread Spectrum Clock Generator of PLL3, in order to reduce the amount of EMI peaks.
0: SSCG disabled (default after reset)
1: SSCG enabled

Bit 23 **PLL3VCOSEL**: PLL3 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL3.
This bit must be written before enabling the PLL3. It allows the application to select the VCO range:
- VCOH: working from 384 to 1672 MHz (Fref2_ck must be between 2 and 16 MHz)
- VCOL: working from 150 to 420 MHz (Fref2_ck must be between 1 and 2 MHz)
0: VCOH selected (default after reset)
1: VCOL selected

Bit 22 **PLL3FRACEN**: PLL3 fractional latch enable
Set and reset by software to latch the content of FRACN into the sigma-delta modulator.
In order to latch the FRACN value into the sigma-delta modulator, PLL3FRACEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN into the modulator.
Refer to **PLL initialization procedure on page 418** for additional information.
PLL3FRACEN must remain at 0 for at least 5 µs

Bit 21 Reserved, must be kept at reset value.

Bit 20 **PLL2TEN**: PLL2 DIVT divider output enable
Set and reset by software to enable the pll2_t_ck output of the PLL2.
To save power, PLL2TEN must be set to 0 when the pll2_t_ck is not used.
The hardware prevents writing this bit if XSPICKP = 1.
0: pll2_t_ck output disabled (default after reset)
1: pll2_t_ck output enabled

Bit 19 **PLL2SEN**: PLL2 DIVS divider output enable
Set and reset by software to enable the pll2_s_ck output of the PLL2.
To save power, PLL2SEN must be set to 0 when the pll2_s_ck is not used.
The hardware prevents writing this bit if XSPICKP = 1.
0: pll2_s_ck output disabled (default after reset)
1: pll2_s_ck output enabled

Bit 18 **PLL2REN**: PLL2 DIVR divider output enable
Set and reset by software to enable the pll2_r_ck output of the PLL2.
The hardware prevents writing this bit if FMCCKP = 1.
To save power, PLL3REN and DIVR bits must be set to 0 when the pll3_r_ck is not used.
0: pll2_r_ck output disabled (default after reset)
1: pll2_r_ck output enabled

Bit 17 **PLL2QEN**: PLL2 DIVQ divider output enable
Set and reset by software to enable the pll2_q_ck output of the PLL2.
To save power, PLL3QEN and DIVQ bits must be set to 0 when the pll3_q_ck is not used.
0: pll2_q_ck output disabled (default after reset)
1: pll2_q_ck output enabled

Bit 16 **PLL2PEN**: PLL2 DIVP divider output enable
Set and reset by software to enable the pll2_p_ck output of the PLL2.
To save power, PLL2PEN and DIVP bits must be set to 0 when the pll2_p_ck is not used.
0: pll2_p_ck output disabled (default after reset)
1: pll2_p_ck output enabled
Bits 15:14  **PLL2RGE[1:0]**: PLL2 input frequency range
Set and reset by software to select the proper reference frequency range used for PLL2.
These bits must be written before enabling the PLL2.
00: PLL3 input (ref2_ck) clock range frequency between 1 and 2 MHz (default after reset)
01: PLL3 input (ref2_ck) clock range frequency between 2 and 4 MHz
10: PLL3 input (ref2_ck) clock range frequency between 4 and 8 MHz
11: PLL3 input (ref2_ck) clock range frequency between 8 and 16 MHz

Bit 13  **PLL2SSCGEN**: PLL2 SSCG enable
Set and reset by software to enable the Spread Spectrum Clock Generator of PLL2, in order to reduce the amount of EMI peaks.
0: SSCG disabled (default after reset)
1: SSCG enabled

Bit 12  **PLL2VCOSEL**: PLL2 VCO selection
Set and reset by software to select the proper VCO frequency range used for PLL2.
This bit must be written before enabling the PLL2. It allows the application to select the VCO range:
- **VCOH**: working from 384 to 1672 MHz (F_{ref2_ck} must be between 2 and 16 MHz)
- **VCOL**: working from 150 to 420 MHz (F_{ref2_ck} must be between 1 and 2 MHz)
0: VCOH selected (default after reset)
1: VCOL selected

Bit 11  **PLL2FRACLEN**: PLL2 fractional latch enable
Set and reset by software to latch the content of FRACN into the sigma-delta modulator.
In order to latch the FRACN value into the sigma-delta modulator, PLL2FRACLEN must be set to 0, then set to 1. The transition 0 to 1 transfers the content of FRACN into the modulator.
Refer to **PLL initialization procedure on page 418** for additional information.
PLL2FRACLEN must remain at 0 for at least 5 µs.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8  **PLL1SEN**: PLL1 DIVS divider output enable
Set and reset by software to enable the pll1_s_ck output of the PLL1.
To save power, PLL1SEN must be set to 0 when the pll1_s_ck is not used.
0: pll1_s_ck output disabled (default after reset)
1: pll1_s_ck output enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6  **PLL1QEN**: PLL1 DIVQ divider output enable
Set and reset by software to enable the pll1_q_ck output of the PLL1.
The hardware prevents writing this bit if FMCCKP = 1.
In order to save power, when the pll1_q_ck output of the PLL1 is not used, the pll1_q_ck must be disabled.
0: pll1_q_ck output disabled (default after reset)
1: pll1_q_ck output enabled

Bit 5  **PLL1PEN**: PLL1 DIVP divider output enable
Set and reset by software to enable the pll1_p_ck output of the PLL1.
The hardware prevents writing this bit to 0, if the PLL1 output is used as the system clock (SW=3).
In order to save power, when the pll1_p_ck output of the PLL1 is not used, the pll1_p_ck must be disabled.
0: pll1_p_ck output disabled (default after reset)
1: pll1_p_ck output enabled
**7.8.12 RCC PLL1 dividers configuration register 1 (RCC_PLL1DIVR1)**

Address offset: 0x030

Reset value: 0x0101 0280

| Bit 31:24 | Reserved, must be kept at reset value. |
| Bit 23  | Reserved, must be kept at reset value. |

![Image of table showing register configuration](image-url)
Bits 22:16 **DIVQ[6:0]**: PLL1 DIVQ division factor
Set and reset by software to control the frequency of the \(\text{pll1\_q\_ck} \) clock.
These bits can be written only when the PLL1QEN = 0.
0000000: \(\text{pll1\_q\_ck} = vco1\_ck\)
0000001: \(\text{pll1\_q\_ck} = vco1\_ck \div 2\) (default after reset)
0000010: \(\text{pll1\_q\_ck} = vco1\_ck \div 3\)
0000011: \(\text{pll1\_q\_ck} = vco1\_ck \div 4\)
... 1111111: \(\text{pll1\_q\_ck} = vco1\_ck \div 128\)

Bits 15:9 **DIVP[6:0]**: PLL1 DIVP division factor
Set and reset by software to control the frequency of the \(\text{pll1\_p\_ck} \) clock.
These bits can be written only when the PLL1PEN = 0.
0000000: \(\text{pll1\_p\_ck} = vco1\_ck\)
0000001: \(\text{pll1\_p\_ck} = vco1\_ck \div 2\) (default after reset)
0000010: not allowed
0000011: \(\text{pll1\_p\_ck} = vco1\_ck \div 4\)
... 1111111: \(\text{pll1\_p\_ck} = vco1\_ck \div 128\)

Bits 8:0 **DIVN[8:0]**: multiplication factor for PLL1 VCO
Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL1ON = PLL1RDN = 0).
.........: not used
0x006: wrong configuration
0x007: DIVN = 8
...
0x080: DIVN = 129 (default after reset)
...
0x1A3: DIVN = 420
Others: wrong configurations

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:
- 128 to 544MHz if PLL1VCOSEL = 0
- 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = \(F_{\text{ref1\_ck}} \times \text{DIVN}\), when fractional value 0 has been loaded into FRACN, with:
- DIVN between 8 and 420
- The input frequency \(F_{\text{ref1\_ck}}\) must be between 1 and 16 MHz.
7.8.13 RCC PLL1 dividers configuration register 2 (RCC_PLL1DIVR2)

Address offset: 0x0C0
Reset value: 0x0000 0101

<table>
<thead>
<tr>
<th>Bits 31:3</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits 2:0</th>
<th>DIVS[2:0]: PLL1 DIVS division factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software to control the frequency of the pll1_s_ck clock.</td>
</tr>
<tr>
<td></td>
<td>This post-divider performs divisions with 50% duty-cycle.</td>
</tr>
<tr>
<td></td>
<td>The duty-cycle of 50% is guaranteed only in the following conditions:</td>
</tr>
<tr>
<td></td>
<td>– With VCOL, if (DIVS+1) is even,</td>
</tr>
<tr>
<td></td>
<td>– With VCOH, for all DIVS values</td>
</tr>
<tr>
<td></td>
<td>These bits can be written only when the PLL1SEN = 0.</td>
</tr>
<tr>
<td>000:</td>
<td>pll1_s_ck = vco1_ck</td>
</tr>
<tr>
<td>001:</td>
<td>pll1_s_ck = vco1_ck / 2 (default after reset)</td>
</tr>
<tr>
<td>010:</td>
<td>pll1_s_ck = vco1_ck / 3</td>
</tr>
<tr>
<td>011:</td>
<td>pll1_s_ck = vco1_ck / 4</td>
</tr>
<tr>
<td>100:</td>
<td>pll1_s_ck = vco1_ck / 5</td>
</tr>
<tr>
<td>101:</td>
<td>pll1_s_ck = vco1_ck / 6</td>
</tr>
<tr>
<td>110:</td>
<td>pll1_s_ck = vco1_ck / 7</td>
</tr>
<tr>
<td>111:</td>
<td>pll1_s_ck = vco1_ck / 8</td>
</tr>
</tbody>
</table>

7.8.14 RCC PLL1 fractional divider register (RCC_PLL1FRACR)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:3</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits 2:0</th>
<th>FRACN[12:0]: Fractional division factor</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software to control the frequency of the pll1_s_ck clock.</td>
</tr>
<tr>
<td></td>
<td>This post-divider performs divisions with 50% duty-cycle.</td>
</tr>
<tr>
<td></td>
<td>The duty-cycle of 50% is guaranteed only in the following conditions:</td>
</tr>
<tr>
<td></td>
<td>– With VCOL, if (DIVS+1) is even,</td>
</tr>
<tr>
<td></td>
<td>– With VCOH, for all DIVS values</td>
</tr>
<tr>
<td></td>
<td>These bits can be written only when the PLL1SEN = 0.</td>
</tr>
<tr>
<td>000:</td>
<td>pll1_s_ck = vco1_ck</td>
</tr>
<tr>
<td>001:</td>
<td>pll1_s_ck = vco1_ck / 2 (default after reset)</td>
</tr>
<tr>
<td>010:</td>
<td>pll1_s_ck = vco1_ck / 3</td>
</tr>
<tr>
<td>011:</td>
<td>pll1_s_ck = vco1_ck / 4</td>
</tr>
<tr>
<td>100:</td>
<td>pll1_s_ck = vco1_ck / 5</td>
</tr>
<tr>
<td>101:</td>
<td>pll1_s_ck = vco1_ck / 6</td>
</tr>
<tr>
<td>110:</td>
<td>pll1_s_ck = vco1_ck / 7</td>
</tr>
<tr>
<td>111:</td>
<td>pll1_s_ck = vco1_ck / 8</td>
</tr>
</tbody>
</table>
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:3  **FRACN[12:0]**: fractional part of the multiplication factor for PLL1 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL1 VCO.

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 544 MHz if PLL1VCOSEL = 0
- 150 to 420 MHz if PLL1VCOSEL = 1

VCO output frequency = \( F_{ref1,ck} \times (DIVN + (FRACN / 2^{13})) \), with
- \( DIVN \) between 8 and 420
- \( FRACN \) can be between 0 and \( 2^{13} - 1 \)
- The input frequency \( F_{ref1,ck} \) must be between 1 and 16 MHz.

To change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0.
- Write the new fractional value into FRACN.
- Set the bit PLL1FRACEN to 1
- Wait at least 5 \( \mu s \).

Bits 2:0  Reserved, must be kept at reset value.
7.8.15 RCC PLL1 Spread Spectrum Clock Generator register (RCC_PLL1SSCGR)

Address offset: 0x0CC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 30:16</td>
<td>INCSTEP[14:0]: Modulation Depth Adjustment for PLL1</td>
</tr>
<tr>
<td>Set and reset by software to adjust the modulation depth of the clock spreading generator.</td>
<td></td>
</tr>
</tbody>
</table>

| Bit 15  | SPREADSEL: Spread spectrum clock generator mode for PLL1 |
| Set and reset by software to select the clock spreading mode. |
| 0: Center-spread modulation selected (default after reset) |
| 1: Down-spread modulation selected |

| Bit 14  | RPDPFNDS: Dithering RPDP noise control for PLL1 |
| Set and reset by software. |
| This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function. |
| 0: Dithering noise injection enabled (default after reset) |
| 1: Dithering noise injection disabled |

| Bit 13  | TPDFNDS: Dithering TPDF noise control for PLL1 |
| Set and reset by software. |
| This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function. |
| 0: Dithering noise injection enabled (default after reset) |
| 1: Dithering noise injection disabled |

| Bits 12:0 MODPER[12:0]: Modulation Period Adjustment for PLL1 |
| Set and reset by software to adjust the modulation period of the clock spreading generator. |
7.8.16 RCC PLL2 dividers configuration register 1 (RCC_PLL2DIVR1)

Address offset: 0x038
Reset value: 0x0101 0280

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>DIVR[6:0]</th>
<th>Bit 24</th>
<th>DIVQ[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **DIVR[6:0]**: PLL2 DIVR division factor

- Set and reset by software to control the frequency of the pll2_r_ck clock.
- These bits can be written only when the PLL2REN = 0.
- 0000000: pll2_r_ck = vco2_ck
- 0000001: pll2_r_ck = vco2_ck / 2 (default after reset)
- 0000010: pll2_r_ck = vco2_ck / 3
- 0000011: pll2_r_ck = vco2_ck / 4
- ...
- 1111111: pll2_r_ck = vco2_ck / 128

Bit 23 Reserved, must be kept at reset value.
Bits 22:16 **DIVQ[6:0]**: PLL2 DIVQ division factor
Set and reset by software to control the frequency of the \texttt{pll2_q_ck} clock.
These bits can be written only when the PLL2QEN = 0.
- 000000: \texttt{pll2_q_ck} = \texttt{vco2_ck}
- 000001: \texttt{pll2_q_ck} = \texttt{vco2_ck} / 2 (default after reset)
- 000010: \texttt{pll2_q_ck} = \texttt{vco2_ck} / 3
- 000011: \texttt{pll2_q_ck} = \texttt{vco2_ck} / 4
- ...
- 111111: \texttt{pll2_q_ck} = \texttt{vco2_ck} / 128

Bits 15:9 **DIVP[6:0]**: PLL2 DIVP division factor
Set and reset by software to control the frequency of the \texttt{pll2_p_ck} clock.
These bits can be written only when the PLL2PEN = 0.
- 000000: \texttt{pll2_p_ck} = \texttt{vco2_ck}
- 000001: \texttt{pll2_p_ck} = \texttt{vco2_ck} / 2 (default after reset)
- 000010: \texttt{pll2_p_ck} = \texttt{vco2_ck} / 3
- 000011: \texttt{pll2_p_ck} = \texttt{vco2_ck} / 4
- ...
- 111111: \texttt{pll2_p_ck} = \texttt{vco2_ck} / 128

Bits 8:0 **DIVN[8:0]**: multiplication factor for PLL2 VCO
Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL2ON = PLL2RDY = 0).
- ..........: not used
- 0x006: wrong configuration
- 0x007: DIVN = 8
- ...
- 0x080: DIVN = 129 (default after reset)
- ...
- 0x1A3: DIVN = 420
- Others: wrong configurations

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:
- 128 to 544 MHz if PLL2VCOSSEL = 0
- 150 to 420 MHz if PLL2VCOSSEL = 1

VCO output frequency = \(F_{\text{ref2}} \times \text{DIVN}\), when fractional value 0 has been loaded into FRACN, with
- \text{DIVN} between 8 and 420
- The input frequency \(F_{\text{ref2}}\) must be between 1 and 16MHz.
7.8.17 **RCC PLL2 dividers configuration register 2 (RCC_PLL2DIVR2)**

Address offset: 0x0C4

Reset value: 0x0000 0101

<table>
<thead>
<tr>
<th>Bits 31:11</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits 10:8</th>
<th>DIVT[2:0]: PLL2 DIVT division factor</th>
</tr>
</thead>
</table>

Set and reset by software to control the frequency of the pll2_t_ck clock.
This post-divider performs divisions with 50% duty-cycle.
The duty-cycle of 50% is guaranteed only in the following conditions:
- With VCOL, if (DIVT+1) is even,
- With VCOH, for all DIVT values

These bits can be written only when the PLL2TEN = 0.

<table>
<thead>
<tr>
<th>000: pll2_t_ck = vco2_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td>001: pll2_t_ck = vco2_ck / 2 (default after reset)</td>
</tr>
<tr>
<td>010: pll2_t_ck = vco2_ck / 3</td>
</tr>
<tr>
<td>011: pll2_t_ck = vco2_ck / 4</td>
</tr>
<tr>
<td>100: pll2_t_ck = vco2_ck / 5</td>
</tr>
<tr>
<td>101: pll2_t_ck = vco2_ck / 6</td>
</tr>
<tr>
<td>110: pll2_t_ck = vco2_ck / 7</td>
</tr>
<tr>
<td>111: pll2_t_ck = vco2_ck / 8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 7:3</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits 2:0</th>
<th>DIVS[2:0]: PLL2 DIVS division factor</th>
</tr>
</thead>
</table>

Set and reset by software to control the frequency of the pll2_s_ck clock.
This post-divider performs divisions with 50% duty-cycle.
The duty-cycle of 50% is guaranteed only in the following conditions:
- With VCOL, if (DIVS+1) is even,
- With VCOH, for all DIVS values

These bits can be written only when the PLL2SEN = 0.

<table>
<thead>
<tr>
<th>000: pll2_s_ck = vco2_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td>001: pll2_s_ck = vco2_ck / 2 (default after reset)</td>
</tr>
<tr>
<td>010: pll2_s_ck = vco2_ck / 3</td>
</tr>
<tr>
<td>011: pll2_s_ck = vco2_ck / 4</td>
</tr>
<tr>
<td>100: pll2_s_ck = vco2_ck / 5</td>
</tr>
<tr>
<td>101: pll2_s_ck = vco2_ck / 6</td>
</tr>
<tr>
<td>110: pll2_s_ck = vco2_ck / 7</td>
</tr>
<tr>
<td>111: pll2_s_ck = vco2_ck / 8</td>
</tr>
</tbody>
</table>
7.8.18  RCC PLL2 fractional divider register (RCC_PLL2FRACR)

Address offset: 0x03C
Reset value: 0x0000 0000

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:3  **FRACN[12:0]**: fractional part of the multiplication factor for PLL2 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO. These bits can be written at any time, allowing dynamic fine-tuning of the PLL2 VCO.

**Caution:** The software must correctly set these bits to ensure that the VCO output frequency is within its valid frequency range, that is:

- 128 to 544 MHz if PLL2VCOSEL = 0
- 150 to 420 MHz if PLL2VCOSEL = 1

VCO output frequency = \( F_{\text{ref2,ck}} \times (\text{DIVN} + \frac{\text{FRACN}}{2^{13}}) \), with

- DIVN between 8 and 420
- FRACN can be between 0 and \( 2^{13} - 1 \)
- The input frequency \( F_{\text{ref2,ck}} \) must be between 1 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL2FRACLEN to 0.
- Write the new fractional value into FRACN.
- Set the bit PLL2FRACLEN to 1
- Wait at least 5 µs.

Bits 2:0  Reserved, must be kept at reset value.
### 7.8.19 RCC PLL2 Spread Spectrum Clock Generator register (RCC_PLL2SSCGR)

Address offset: 0x0D0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>30:16</td>
<td>INCSTEP[14:0]: Modulation Depth Adjustment for PLL2</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>SPREADSEL: Spread spectrum clock generator mode for PLL2</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>RPDFNDIS: Dithering RPDF noise control for PLL2</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>TPDFNDIS: Dithering TPDF noise control for PLL2</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>12:0</td>
<td>MODPER[12:0]: Modulation Period Adjustment for PLL2</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

- **INCSTEP[14:0]**: Modulation Depth Adjustment for PLL2
  - Set and reset by software to adjust the modulation depth of the clock spreading generator.

- **SPREADSEL**: Spread spectrum clock generator mode for PLL2
  - Set and reset by software to select the clock spreading mode.
  - 0: Center-spread modulation selected (default after reset)
  - 1: Down-spread modulation selected

- **RPDFNDIS**: Dithering RPDF noise control for PLL2
  - Set and reset by software.
  - This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.
  - 0: Dithering noise injection enabled (default after reset)
  - 1: Dithering noise injection disabled

- **TPDFNDIS**: Dithering TPDF noise control for PLL2
  - Set and reset by software.
  - This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.
  - 0: Dithering noise injection enabled (default after reset)
  - 1: Dithering noise injection disabled

- **MODPER[12:0]**: Modulation Period Adjustment for PLL2
  - Set and reset by software to adjust the modulation period of the clock spreading generator.
7.8.20  RCC PLL3 dividers configuration register 1 (RCC_PLL3DIVR1)

Address offset: 0x040
Reset value: 0x0101 0280

Bit 31  Reserved, must be kept at reset value.

Bits 30:24  DIVR[6:0]: PLL3 DIVR division factor
- Set and reset by software to control the frequency of the pll3_r_ck clock.
- These bits can be written only when the PLL3REN = 0.
  0000000: pll3_r_ck = vco3_ck
  0000001: pll3_r_ck = vco3_ck / 2 (default after reset)
  0000010: pll3_r_ck = vco3_ck / 3
  0000011: pll3_r_ck = vco3_ck / 4
  ...
  1111111: pll3_r_ck = vco3_ck / 128

Bit 23  Reserved, must be kept at reset value.
Bits 22:16 **DIVQ[6:0]**: PLL3 DIVQ division factor
Set and reset by software to control the frequency of the `pll3_q_ck` clock.
These bits can be written only when the PLL3QEN = 0.
- **0000000**: \( \text{pll3}_q\_ck = \text{vco3}_ck \)
- **0000001**: \( \text{pll3}_q\_ck = \text{vco3}_ck / 2 \) (default after reset)
- **0000010**: \( \text{pll3}_q\_ck = \text{vco3}_ck / 3 \)
- **0000011**: \( \text{pll3}_q\_ck = \text{vco3}_ck / 4 \)
- ... 
- **1111111**: \( \text{pll3}_q\_ck = \text{vco3}_ck / 128 \)

Bits 15:9 **DIVP[6:0]**: PLL3 DIVP division factor
Set and reset by software to control the frequency of the `pll3_p_ck` clock.
These bits can be written only when the PLL3PEN = 0.
- **0000000**: \( \text{pll3}_p\_ck = \text{vco3}_ck \)
- **0000001**: \( \text{pll3}_p\_ck = \text{vco3}_ck / 2 \) (default after reset)
- **0000010**: \( \text{pll3}_p\_ck = \text{vco3}_ck / 3 \)
- **0000011**: \( \text{pll3}_p\_ck = \text{vco3}_ck / 4 \)
- ... 
- **1111111**: \( \text{pll3}_p\_ck = \text{vco3}_ck / 128 \)

Bits 8:0 **DIVN[8:0]**: Multiplication factor for PLL3 VCO
Set and reset by software to control the multiplication factor of the VCO.
These bits can be written only when the PLL is disabled (PLL3ON = PLL3RDY = 0).
- ..........: not used
- **0x006**: wrong configuration
- **0x007**: DIVN = 8
- ... 
- **0x080**: DIVN = 129 (default after reset)
- ... 
- **0x1A3**: DIVN = 420
- Others: wrong configurations

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:
- 128 to 544 MHz if PLL3VCOSSEL = 0
- 150 to 420 MHz if PLL3VCOSSEL = 1

VCO output frequency = \( F_{\text{ref3}} \_ck \times \text{DIVN} \), when fractional value 0 has been loaded into FRACN, with:
- DIVN between 8 and 420
- The input frequency \( F_{\text{ref3}} \_ck \) must be between 1 and 16 MHz
### 7.8.21 RCC PLL3 dividers configuration register 2 (RCC_PLL3DIVR2)

**Address offset:** 0x0C8

**Reset value:** 0x0000 0101

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

#### Bits 2:0 **DIVS[2:0]**: PLL3 DIVS division factor

- Set and reset by software to control the frequency of the pll3_s_ck clock.
- This post-divider performs divisions with 50% duty-cycle.
- The duty-cycle of 50% is guaranteed only in the following conditions:
  - With VCOL, if (DIVS+1) is even,
  - With VCOH, for all DIVS values

These bits can be written only when the PLL3SEN = 0.

- 000: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck \)
- 001: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 2 \) (default after reset)
- 010: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 3 \)
- 011: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 4 \)
- 100: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 5 \)
- 101: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 6 \)
- 110: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 7 \)
- 111: \( \text{pll3}_\_s\_ck = \text{vco3}_\_ck / 8 \)

### 7.8.22 RCC PLL3 fractional divider register (RCC_PLL3FRACCR)

**Address offset:** 0x044

**Reset value:** 0x0000 0000

#### Bits 31:16
Bits 15:13 Reserved, must be kept at reset value.

#### Bits 12:0 **FRACN[12:0]**

- These bits can be written only when the PLL3SEN = 0.
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:3  **FRACN[12:0]**: fractional part of the multiplication factor for PLL3 VCO

Set and reset by software to control the fractional part of the multiplication factor of the VCO.

These bits can be written at any time, allowing dynamic fine-tuning of the PLL3 VCO.

**Caution:** The software must set correctly these bits to insure that the VCO output frequency is between its valid frequency range, that is:

- 128 to 544 MHz if PLL3VCOSEL = 0
- 150 to 420 MHz if PLL3VCOSEL = 1

VCO output frequency = \( F_{\text{ref3,ck}} \times (\text{DIVN} + (\text{FRACN} / 2^{13})) \), with

- DIVN between 8 and 420
- FRACN can be between 0 and \( 2^{13} - 1 \)
- The input frequency \( F_{\text{ref3,ck}} \) must be between 1 and 16 MHz.

In order to change the FRACN value on-the-fly even if the PLL is enabled, the application must proceed as follows:

- Set the bit PLL1FRACEN to 0.
- Write the new fractional value into FRACN.
- Set the bit PLL1FRACEN to 1
- Wait at least 5 \( \mu \)s.

Bits 2:0  Reserved, must be kept at reset value.
7.8.23  **RCC PLL3 Spread Spectrum Clock Generator register**  
(RCC_PLL3SSCGR)  
Address offset: 0x0D4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30-16</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCSTEP[14:0]</td>
<td></td>
<td>SPREADSEL</td>
<td>RPDFNDIS</td>
<td>TPDFNDIS</td>
<td>MODPER[12:0]</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Bit 31**: Reserved, must be kept at reset value.
- **Bits 30:16** **INCSTEP[14:0]**: Modulation Depth Adjustment for PLL3  
  Set and reset by software to adjust the modulation depth of the clock spreading generator.
- **Bit 15** **SPREADSEL**: Spread spectrum clock generator mode for PLL3  
  Set and reset by software to select the clock spreading mode.  
  - 0: Center-spread modulation selected (default after reset)  
  - 1: Down-spread modulation selected
- **Bit 14** **RPDFNDIS**: Dithering RPDF noise control for PLL3  
  Set and reset by software.  
  This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a rectangular probability density function.  
  - 0: Dithering noise injection enabled (default after reset)  
  - 1: Dithering noise injection disabled
- **Bit 13** **TPDFNDIS**: Dithering TPDF noise control for PLL3  
  Set and reset by software.  
  This bit is used to enable or disable the injection of a dithering noise into the SSCG modulator. This dithering noise is generated using a triangular probability density function.  
  - 0: Dithering noise injection enabled (default after reset)  
  - 1: Dithering noise injection disabled
- **Bits 12:0** **MODPER[12:0]**: Modulation Period Adjustment for PLL3  
  Set and reset by software to adjust the modulation period of the clock spreading generator.
### 7.8.24 RCC AHB peripheral kernel clock selection register (RCC_CCIPR1)

Address offset: 0x04C

Reset value: 0x0000 0A00

Changing the clock source on-the-fly is allowed (except for CKPERSEL and ETH1REFCKSEL[1:0]) and does not generate any timing violation. However, the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to *Clock switches and gating*.

![Register Table](image)

Bits 31:30: Reserved, must be kept at reset value.

Bits 29:28: **CKPERSEL[1:0]: per_ck clock source selection**

- 00: `hsi_ker_ck` selected as `per_ck` clock (default after reset)
- 01: `csi_ker_ck` selected as `per_ck` clock
- 10: `hse_ker_ck` selected as `per_ck` clock
- 11: reserved, the `ck_per` clock is disabled

Bit 27: **PSSISEL**: PSSI kernel clock source selection

Set and reset by software.

- 0: `pll3_r_ck` selected as kernel peripheral clock (default after reset)
- 1: `per_ck` selected as kernel peripheral clock

Bit 26: Reserved, must be kept at reset value.

Bits 25:24: **ADCSEL[1:0]: SAR ADC kernel clock source selection**

Set and reset by software.

- 00: `pll2_p_ck` selected as kernel peripheral clock (default after reset)
- 01: `pll3_r_ck` selected as kernel peripheral clock
- 10: `per_ck` selected as kernel peripheral clock
- others: reserved, the kernel clock is disabled

Bit 23: Reserved, must be kept at reset value.
Bits 22:20  ADF1SEL[2:0]: ADF kernel clock source selection
   Set and reset by software.
   0: hclk1 selected as ADF kernel clock (default after reset)
   1: pll2_p_ck selected as ADF kernel clock
   2: pll3_p_ck selected as ADF kernel clock
   3: I2S_CKIN selected as ADF kernel clock
   4: csi_ker_ck selected as ADF kernel clock
   5: hsi_ker_ck selected as ADF kernel clock
   Note: I2S_CKIN is an external clock taken from a pin.

Bit 19  Reserved, must be kept at reset value.

Bit 18  ETH1PHYCKSEL: Clock source selection for external Ethernet PHY
   Set and reset by software.
   0: hse_ker_ck selected as clock source (default after reset)
   1: pll3_s_ck selected clock source

Bits 17:16  ETH1REFCKSEL[1:0]: Ethernet reference clock source selection
   Set and reset by software.
   00: PAD ETH_RMII_REF_CLK selected as kernel peripheral clock (default after reset)
   01: hse_ker_ck selected as kernel peripheral clock
   10: eth_clk_fb selected as kernel peripheral clock
   others: reserved, the kernel clock is disabled

Bits 15:14  OTGFSSEL[1:0]: OTGFS kernel clock source selection
   Set and reset by software.
   00: hsi48_ker_ck (default after reset)
   01: pll3_q_ck
   10: hse_ker_ck
   11: clk48mohci

Bits 13:12  USBPHYCSEL[1:0]: USBPHYC kernel clock source selection
   Set and reset by software.
   00: hse_ker_ck (default after reset)
   01: hse_ker_ck / 2
   10: pll3_q_ck
   11: reserved, the kernel clock is disabled

Bits 11:8  USBREFCKSEL[3:0]: USBPHYC kernel clock frequency selection
   Set and reset by software.
   This field is used to indicate to the USBPHYC, the frequency of the reference kernel clock provided
   to the USBPHYC.
   0011: The kernel clock frequency provided to the USBPHYC is 16 MHz
   1000: The kernel clock frequency provided to the USBPHYC is 19.2 MHz
   1001: The kernel clock frequency provided to the USBPHYC is 20 MHz
   1010: The kernel clock frequency provided to the USBPHYC is 24 MHz (default after reset)
   1110: The kernel clock frequency provided to the USBPHYC is 32 MHz
   others: reserved

Bits 7:6  XSPI2SEL[1:0]: XSPI2 kernel clock source selection
   Set and reset by software.
   00: hclk5 selected as kernel peripheral clock (default after reset)
   01: pll2_s_ck selected as kernel peripheral clock
   1x: pll2_t_ck selected as kernel peripheral clock
Bits 5:4  **XSPI1SEL[1:0]**: XSPI1 kernel clock source selection  
Set and reset by software.  
- 00: **hclk5** selected as kernel peripheral clock (default after reset)  
- 01: **pll2_s_ck** selected as kernel peripheral clock  
- 1x: **pll2_t_ck** selected as kernel peripheral clock

Bit 3  Reserved, must be kept at reset value.

Bit 2  **SDMMC12SEL**: SDMMC1 and SDMMC2 kernel clock source selection  
Set and reset by software.  
- 0: **pll2_s_ck** selected as kernel peripheral clock (default after reset)  
- 1: **pll2_t_ck** selected as kernel peripheral clock

Bits 1:0  **FMCSEL[1:0]**: FMC kernel clock source selection  
Set and reset by software.  
- 00: **hclk5** selected as kernel peripheral clock (default after reset)  
- 01: **pll1_q_ck** selected as kernel peripheral clock  
- 10: **pll2_r_ck** selected as kernel peripheral clock  
- 11: **hsi_ker_ck** selected as kernel peripheral clock
7.8.25 RCC APB1 peripherals kernel clock selection register (RCC_CCIPR2)

Address offset: 0x050
Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to *Clock switches and gating*.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LPTIM1SEL[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CECSEL[1:0]**: HDMI-CEC kernel clock source selection
Set and reset by software.
00: lse_ck selected as kernel clock (default after reset)
01: lsi_ck selected as kernel clock
10: csi_ker_ck divided by 122 selected as kernel clock
11: reserved, the kernel clock is disabled

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **SPDIFRXSEL[1:0]**: SPDIFRX kernel clock source selection
00: pll1_q_ck selected as SPDIFRX kernel clock (default after reset)
01: pll2_r_ck selected as SPDIFRX kernel clock
10: pll3_r_ck selected as SPDIFRX kernel clock
11: hsi_ker_ck selected as SPDIFRX kernel clock

Bits 23:22 **FDCANSEL[1:0]**: FDCAN kernel clock source selection
00: hse_ker_ck selected as FDCAN kernel clock (default after reset)
01: pll1_q_ck selected as FDCAN kernel clock
10: pll2_p_ck selected as FDCAN kernel clock
11: reserved, the kernel clock is disabled

Bits 21:19 Reserved, must be kept at reset value.
RM0477  Reset and clock control (RCC)

Bits 18:16  **LPTIM1SEL[2:0]**: LPTIM1 kernel clock source selection
Set and reset by software.
- 000: pclk1 selected as kernel peripheral clock (default after reset)
- 001: pll2_p_ck selected as kernel peripheral clock
- 010: pll3_r_ck selected as kernel peripheral clock
- 011: lse_ck selected as kernel peripheral clock
- 100: lsi_ck selected as kernel peripheral clock
- 101: per_ck selected as kernel peripheral clock
others: reserved, the kernel clock is disabled

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

Bits 13:12  **I2C1_I3C1SEL[1:0]**: I2C1 or I3C1 kernel clock source selection
Set and reset by software.
- 00: pclk1 selected as kernel peripheral clock (default after reset)
- 01: pll3_r_ck selected as kernel peripheral clock
- 10: hsi_ker_ck selected as kernel peripheral clock
- 11: csi_ker_ck selected as kernel peripheral clock

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:8  **I2C23SEL[1:0]**: I2C2, I2C3 kernel clock source selection
Set and reset by software.
- 00: pclk1 selected as kernel clock (default after reset)
- 01: pll3_r_ck selected as kernel clock
- 10: hsi_ker_ck selected as kernel clock
- 11: csi_ker_ck selected as kernel clock

Bit 7  Reserved, must be kept at reset value.

Bits 6:4  **SPI23SEL[2:0]**: SPI/I2S2 and SPI/I2S3 kernel clock source selection
Set and reset by software.
Caution: If the selected clock is the external clock and this clock is stopped, it is not be possible to switch to another clock. Refer to Clock switches and gating on page 411 for additional information.
- 000: pll1_q_ck selected as kernel clock (default after reset)
- 001: pll2_p_ck selected as kernel clock
- 010: pll3_p_ck selected as kernel clock
- 011: I2S_CKIN selected as kernel clock
- 100: per_ck selected as kernel clock
others: reserved, the kernel clock is disabled
Note: I2S_CKIN is an external clock taken from a pin.

Bits 5:3  Reserved, must be kept at reset value.

Bits 2:0  **UART234578SEL[2:0]**: USART2,3, UART4,5,7,8 (APB1) kernel clock source selection
Set and reset by software.
- 000: pclk1 selected as kernel clock (default after reset)
- 001: pll2_q_ck selected as kernel clock
- 010: pll3_q_ck selected as kernel clock
- 011: hsi_ker_ck selected as kernel clock
- 100: csi_ker_ck selected as kernel clock
- 101: lse_ck selected as kernel clock
others: reserved, the kernel clock is disabled
7.8.26 RCC APB2 peripherals kernel clock selection register (RCC_CCIPR3)

Address offset: 0x054
Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to Clock switches and gating.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 22:20 **SAI2SEL[2:0]**: SAI2 kernel clock source selection
Set and reset by software.

**Caution:** If the selected clock is the external clock and this clock is stopped, it is not possible to switch to another clock. Refer to Clock switches and gating on page 411 for additional information.

- 000: pll1_q_ck selected as SAI2 kernel clock (default after reset)
- 001: pll2_p_ck selected as SAI2 kernel clock
- 010: pll3_p_ck selected as SAI2 kernel clock
- 011: I2S_CKIN selected as SAI2 kernel clock
- 100: per_ck selected as SAI2 kernel clock
- 101: spdifrx_symb_ck selected as SAI2 kernel clock
- Others: reserved, the kernel clock is disabled

**Note:** I2S_CKIN is an external clock taken from a pin. spdifrx_symb_ck is the symbol clock generated by the spdifrx (see Figure 53).

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **SAI1SEL[2:0]**: SAI1 kernel clock source selection
Set and reset by software.

**Caution:** If the selected clock is the external clock and this clock is stopped, it is not possible to switch to another clock. Refer to Clock switches and gating on page 411 for additional information.

- 000: pll1_q_ck selected as SAI1 kernel clock (default after reset)
- 001: pll2_p_ck selected as SAI1 kernel clock
- 010: pll3_p_ck selected as SAI1 kernel clock
- 011: I2S_CKIN selected as SAI1 kernel clock
- 100: per_ck selected as SAI1 kernel clock
- Others: reserved, the kernel clock is disabled

**Note:** I2S_CKIN is an external clock taken from a pin.

Bits 15:11 Reserved, must be kept at reset value.
Bits 10:8 **SPI1SEL[2:0]**: SPI/I2S1 kernel clock source selection
Set and reset by software.

**Caution:** If the selected clock is the external clock and this clock is stopped, it is not be possible to switch to another clock. Refer to Clock switches and gating on page 411 for additional information.

000: **pll1_q_ck** selected as SPI/I2S1 and 7 kernel clock (default after reset)
001: **pll2_p_ck** selected as SPI/I2S1 and 7 kernel clock
010: **pll3_p_ck** selected as SPI/I2S1 and 7 kernel clock
011: **I2S_CKIN** selected as SPI/I2S1 and 7 kernel clock
100: **per_ck** selected as SPI/I2S1, and 7 kernel clock
others: reserved, the kernel clock is disabled

*Note: I2S_CKIN is an external clock taken from a pin.*

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SPI45SEL[2:0]**: SPI4 and 5 kernel clock source selection
Set and reset by software.

000: **pclk2** selected as kernel clock (default after reset)
001: **pll2_q_ck** selected as kernel clock
010: **pll3_q_ck** selected as kernel clock
011: **hsi_ker_ck** selected as kernel clock
100: **csi_ker_ck** selected as kernel clock
101: **hse_ker_ck** selected as kernel clock
others: reserved, the kernel clock is disabled

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **USART1SEL[2:0]**: USART1 kernel clock source selection
Set and reset by software.

000: **pclk2** selected as kernel clock (default after reset)
001: **pll2_q_ck** selected as kernel clock
010: **pll3_q_ck** selected as kernel clock
011: **hsi_ker_ck** selected as kernel clock
100: **csi_ker_ck** selected as kernel clock
101: **lse_ck** selected as kernel clock
others: reserved, the kernel clock is disabled
7.8.27 RCC APB4,5 peripherals kernel clock selection register (RCC_CCIPR4)

Address offset: 0x058
Reset value: 0x0000 0000

Changing the clock source on-the-fly is allowed and does not generate any timing violation. However the user must make sure that both the previous and the new clock sources are present during the switching, and during the whole transition time. Refer to Clock switches and gating.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:12 **LPTIM45SEL[2:0]**: LPTIM4, and LPTIM5 kernel clock source selection
Set and reset by software.
- 000: **pclk4** selected as kernel peripheral clock (default after reset)
- 001: **pll2_p_ck** selected as kernel peripheral clock
- 010: **pll3_r_ck** selected as kernel peripheral clock
- 011: **lse_ck** selected as kernel peripheral clock
- 100: **lsi_ck** selected as kernel peripheral clock
- 101: **per_ck** selected as kernel peripheral clock
- others: reserved, the kernel clock is disabled

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LPTIM23SEL[2:0]**: LPTIM2 and LPTIM3 kernel clock source selection
Set and reset by software.
- 000: **pclk4** selected as kernel peripheral clock (default after reset)
- 001: **pll2_p_ck** selected as kernel peripheral clock
- 010: **pll3_r_ck** selected as kernel peripheral clock
- 011: **lse_ck** selected as kernel peripheral clock
- 100: **lsi_ck** selected as kernel peripheral clock
- 101: **per_ck** selected as kernel peripheral clock
- others: reserved, the kernel clock is disabled

Bit 7 Reserved, must be kept at reset value.
Bits 6:4  **SPI6SEL[2:0]**: SPI/I2S6 kernel clock source selection
   Set and reset by software.
   000: `pclk4` selected as kernel peripheral clock (default after reset)
   001: `pll2_q_ck` selected as kernel peripheral clock
   010: `pll3_q_ck` selected as kernel peripheral clock
   011: `hsi_ker_ck` selected as kernel peripheral clock
   100: `csi_ker_ck` selected as kernel peripheral clock
   101: `hse_ker_ck` selected as kernel peripheral clock
   others: reserved, the kernel clock is disabled

Bit 3  Reserved, must be kept at reset value.

Bits 2:0  **LPUART1SEL[2:0]**: LPUART1 kernel clock source selection
   Set and reset by software.
   000: `pclk4` selected as kernel peripheral clock (default after reset)
   001: `pll2_q_ck` selected as kernel peripheral clock
   010: `pll3_q_ck` selected as kernel peripheral clock
   011: `hsi_ker_ck` selected as kernel peripheral clock
   100: `csi_ker_ck` selected as kernel peripheral clock
   101: `lse_ck` selected as kernel peripheral clock
   others: reserved, the kernel clock is disabled
### 7.8.28 RCC clock source interrupt enable register (RCC_CIER)

**Address offset:** 0x060  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:10 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:10 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9 <strong>LSECSSIE</strong>: LSE clock security system interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by the clock security system (CSS) on external 32 kHz oscillator.</td>
</tr>
<tr>
<td>0: LSE CSS interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: LSE CSS interrupt enabled</td>
</tr>
<tr>
<td>Bit 8 <strong>PLL3RDYIE</strong>: PLL3 ready interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by PLL3 lock.</td>
</tr>
<tr>
<td>0: PLL3 lock interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: PLL3 lock interrupt enabled</td>
</tr>
<tr>
<td>Bit 7 <strong>PLL2RDYIE</strong>: PLL2 ready interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by PLL2 lock.</td>
</tr>
<tr>
<td>0: PLL2 lock interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: PLL2 lock interrupt enabled</td>
</tr>
<tr>
<td>Bit 6 <strong>PLL1RDYIE</strong>: PLL1 ready interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by PLL1 lock.</td>
</tr>
<tr>
<td>0: PLL1 lock interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: PLL1 lock interrupt enabled</td>
</tr>
<tr>
<td>Bit 5 <strong>HSI48RDYIE</strong>: HSI48 ready interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by the HSI48 oscillator stabilization.</td>
</tr>
<tr>
<td>0: HSI48 ready interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: HSI48 ready interrupt enabled</td>
</tr>
<tr>
<td>Bit 4 <strong>CSIRDYIE</strong>: CSI ready interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by the CSI oscillator stabilization.</td>
</tr>
<tr>
<td>0: CSI ready interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: CSI ready interrupt enabled</td>
</tr>
<tr>
<td>Bit 3 <strong>HSERDYIE</strong>: HSE ready interrupt enable</td>
</tr>
<tr>
<td>Set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.</td>
</tr>
<tr>
<td>0: HSE ready interrupt disabled (default after reset)</td>
</tr>
<tr>
<td>1: HSE ready interrupt enabled</td>
</tr>
</tbody>
</table>
### RCC clock source interrupt flag register (RCC_CIFR)

**Address offset:** 0x64  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>HSECSSF</td>
<td>LSECSSF</td>
<td>PLL3RDYF</td>
<td>PLL2RDYF</td>
<td>PLL1RDYF</td>
<td>PLL0RDYF</td>
<td>HSI48RDYF</td>
<td>CSIRDYF</td>
<td>HSERDYF</td>
<td>HSI2RDYF</td>
<td>HSI1RDYF</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

**Bit 10 HSECSSF:** HSE clock security system interrupt flag  
- Reset by software by writing HSECSSC bit.  
- Set by hardware in case of HSE clock failure.  
  0: no clock security interrupt caused by HSE clock failure (default after reset)  
  1: clock security interrupt caused by HSE clock failure

**Bit 9 LSECSSF:** LSE clock security system interrupt flag  
- Reset by software by writing LSECSSC bit.  
- Set by hardware when a failure is detected on the external 32 kHz oscillator and LSECSSIE is set.  
  0: no failure detected on the external 32 kHz oscillator (default after reset)  
  1: failure detected on the external 32 kHz oscillator

**Bit 8 PLL3RDYF:** PLL3 ready interrupt flag  
- Reset by software by writing PLL3RDYC bit.  
- Set by hardware when the PLL3 locks and PLL3RDYIE is set.  
  0: no clock ready interrupt caused by PLL3 lock (default after reset)  
  1: clock ready interrupt caused by PLL3 lock
Bit 7  **PLL2RDYF**: PLL2 ready interrupt flag
Reset by software by writing PLL2RDYC bit.
Set by hardware when the PLL2 locks and PLL2RDYIE is set.
0: no clock ready interrupt caused by PLL2 lock (default after reset)
1: clock ready interrupt caused by PLL2 lock

Bit 6  **PLL1RDYF**: PLL1 ready interrupt flag
Reset by software by writing PLL1RDYC bit.
Set by hardware when the PLL1 locks and PLL1RDYIE is set.
0: no clock ready interrupt caused by PLL1 lock (default after reset)
1: clock ready interrupt caused by PLL1 lock

Bit 5  **HSI48RDYF**: HSI48 ready interrupt flag
Reset by software by writing HSI48RDYC bit.
Set by hardware when the HSI48 clock becomes stable and HSI48RDYIE is set.
0: no clock ready interrupt caused by the HSI48 oscillator (default after reset)
1: clock ready interrupt caused by the HSI48 oscillator

Bit 4  **CSIRDYF**: CSI ready interrupt flag
Reset by software by writing CSIRDYC bit.
Set by hardware when the CSI clock becomes stable and CSIRDYIE is set.
0: no clock ready interrupt caused by the CSI (default after reset)
1: clock ready interrupt caused by the CSI

Bit 3  **HSERDYF**: HSE ready interrupt flag
Reset by software by writing HSERDYC bit.
Set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: no clock ready interrupt caused by the HSE (default after reset)
1: clock ready interrupt caused by the HSE

Bit 2  **HSIRDYF**: HSI ready interrupt flag
Reset by software by writing HSIRDYC bit.
Set by hardware when the HSI clock becomes stable and HSIRDYIE is set.
0: no clock ready interrupt caused by the HSI (default after reset)
1: clock ready interrupt caused by the HSI

Bit 1  **LSERDYF**: LSE ready interrupt flag
Reset by software by writing LSERDYC bit.
Set by hardware when the LSE clock becomes stable and LSERDYIE is set.
0: no clock ready interrupt caused by the LSE (default after reset)
1: clock ready interrupt caused by the LSE

Bit 0  **LSIRDYF**: LSI ready interrupt flag
Reset by software by writing LSIRDYC bit.
Set by hardware when the LSI clock becomes stable and LSIRDYIE is set.
0: no clock ready interrupt caused by the LSI (default after reset)
1: clock ready interrupt caused by the LSI
7.8.30 RCC clock source interrupt clear register (RCC_CICR)

Address offset: 0x68
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:11 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td><strong>HSECSSC</strong>: HSE clock security system interrupt clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set by software to clear HSECSSF.</td>
<td>0:</td>
<td>1:</td>
</tr>
<tr>
<td></td>
<td>Reset by hardware when clear done.</td>
<td>HSECSSF no effect (default after reset)</td>
<td>HSECSSF cleared</td>
</tr>
<tr>
<td>9</td>
<td><strong>LSECSSC</strong>: LSE clock security system interrupt clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set by software to clear LSECSSF.</td>
<td>0:</td>
<td>1:</td>
</tr>
<tr>
<td></td>
<td>Reset by hardware when clear done.</td>
<td>LSECSSF no effect (default after reset)</td>
<td>LSECSSF cleared</td>
</tr>
<tr>
<td>8</td>
<td><strong>PLL3RDYC</strong>: PLL3 ready interrupt clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set by software to clear PLL3RDYF.</td>
<td>0:</td>
<td>1:</td>
</tr>
<tr>
<td></td>
<td>Reset by hardware when clear done.</td>
<td>PLL3RDYF no effect (default after reset)</td>
<td>PLL3RDYF cleared</td>
</tr>
<tr>
<td>7</td>
<td><strong>PLL2RDYC</strong>: PLL2 ready interrupt clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set by software to clear PLL2RDYF.</td>
<td>0:</td>
<td>1:</td>
</tr>
<tr>
<td></td>
<td>Reset by hardware when clear done.</td>
<td>PLL2RDYF no effect (default after reset)</td>
<td>PLL2RDYF cleared</td>
</tr>
<tr>
<td>6</td>
<td><strong>PLL1RDYC</strong>: PLL1 ready interrupt clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set by software to clear PLL1RDYF.</td>
<td>0:</td>
<td>1:</td>
</tr>
<tr>
<td></td>
<td>Reset by hardware when clear done.</td>
<td>PLL1RDYF no effect (default after reset)</td>
<td>PLL1RDYF cleared</td>
</tr>
<tr>
<td>5</td>
<td><strong>HSI48RDYC</strong>: HSI48 ready interrupt clear</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Set by software to clear HSI48RDYF.</td>
<td>0:</td>
<td>1:</td>
</tr>
<tr>
<td></td>
<td>Reset by hardware when clear done.</td>
<td>HSI48RDYF no effect (default after reset)</td>
<td>HSI48RDYF cleared</td>
</tr>
</tbody>
</table>
Bit 4  **CSIRDYC**: CSI ready interrupt clear
       Set by software to clear CSIRDYF.
       Reset by hardware when clear done.
       0: CSIRDYF no effect (default after reset)
       1: CSIRDYF cleared

Bit 3  **HSERDYC**: HSE ready interrupt clear
       Set by software to clear HSERDYF.
       Reset by hardware when clear done.
       0: HSERDYF no effect (default after reset)
       1: HSERDYF cleared

Bit 2  **HSIRDYC**: HSI ready interrupt clear
       Set by software to clear HSIRDYF.
       Reset by hardware when clear done.
       0: HSIRDYF no effect (default after reset)
       1: HSIRDYF cleared

Bit 1  **LSERDYC**: LSE ready interrupt clear
       Set by software to clear LSERDYF.
       Reset by hardware when clear done.
       0: LSERDYF no effect (default after reset)
       1: LSERDYF cleared

Bit 0  **LSIRDYC**: LSI ready interrupt clear
       Set by software to clear LSIRDYF.
       Reset by hardware when clear done.
       0: LSIRDYF no effect (default after reset)
       1: LSIRDYF cleared
### 7.8.31 RCC Backup domain control register (RCC_BDCR)

Address offset: 0x070  
Reset value: 0x0000 0010  
Reset by Backup domain reset.

Access: $0 \leq \text{wait state} \leq 7$, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

After a system reset, the RCC_BDCR register is write-protected. To modify this register, the DBP bit in the PWR control register 1 (PWR_CR1) must be set to 1. RCC_BDCR bits are only reset after a Backup domain reset (see Section 7.4.5: Backup domain reset). Any other internal or external reset does not have any effect on these bits.

#### Table: RCC_BDCR register

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>VSWRST</td>
<td>VSwitch domain software reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software. To generate a VSW reset, it is recommended to write this bit to 1,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>then back to 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: reset not activated (default after Backup domain reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: generates a reset pulse, resetting the entire VSW domain.</td>
</tr>
<tr>
<td>15</td>
<td>RTCEN</td>
<td>RTC clock enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software. 0: rtc_ck disabled (default after Backup domain reset)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: rtc_ck enabled</td>
</tr>
<tr>
<td>14</td>
<td>LSECSSRA</td>
<td>Re-Arm the LSECSS function</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software. After a LSE failure detection, the software application can re-enable the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LSECSS by writing this bit to 1. Reading this bit returns the written value. Prior to set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>this bit to 1, LSECSSON must be set to 0. Refer to Section 7.4.4: CSS on LSE for details.</td>
</tr>
<tr>
<td>13</td>
<td>LSECSSD</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LSECSSON</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>LSEDRV[1:0]</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>LSEBYP</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>LSERDY</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>LSEON</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>RTCSEL[1:0]</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LSEEXT</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LSECKSD</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LSECKSSD</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 16 **VSWRST**: VSwitch domain software reset  
Set and reset by software.  
To generate a VSW reset, it is recommended to write this bit to 1, then back to 0.  
0: reset not activated (default after Backup domain reset)  
1: generates a reset pulse, resetting the entire VSW domain.

Bit 15 **RTCEN**: RTC clock enable  
Set and reset by software.  
0: rtc_ck disabled (default after Backup domain reset)  
1: rtc_ck enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LSECSSRA**: Re-Arm the LSECSS function  
Set by software.  
After a LSE failure detection, the software application can re-enable the LSECSS by writing this bit to 1. Reading this bit returns the written value. Prior to set this bit to 1, LSECSSON must be set to 0. Refer to Section 7.4.4: CSS on LSE for details.  
0: Writing 0 has no effect (default after Backup domain reset)  
1: Writing 1 generates a re-arm pulse for the LSECSS function

Bits 11:10 Reserved, must be kept at reset value.
Bits 9:8 RTCSEL[1:0]: RTC clock source selection
- Set by software to select the clock source for the RTC. These bits can be written only one time (except in case of failure detection on LSE). These bits must be written before LSECSSON is enabled. The VSWRST bit can be used to reset them, then it can be written one time again. If HSE is selected as RTC clock, this clock is lost when the system is in Stop mode or in case of a pin reset (NRST).
  - 00: no clock (default after Backup domain reset)
  - 01: LSE selected as RTC clock
  - 10: LSI selected as RTC clock
  - 11: HSE divided by RTCPRE value selected as RTC clock

Bit 7 LSEEXT: low-speed external clock type in Bypass mode
- Set and reset by software to select the external clock type (analog or digital).
- The external clock must be enabled with the LSEON bit, to be used by the device.
- The LSEEXT bit can be written only if the LSE oscillator is disabled.
  - 0: LSE in analog mode (default after Backup domain reset)
  - 1: LSE in digital mode (do not use if RTC is active)

Bit 6 LSECSSD: LSE clock security system failure detection
- Set by hardware to indicate when a failure has been detected by the clock security system on the external 32 kHz oscillator.
  - 0: no failure detected on 32 kHz oscillator (default after Backup domain reset)
  - 1: failure detected on 32 kHz oscillator

Bit 5 LSECSSON: LSE clock security system enable
- Set by software to enable the clock security system on 32 kHz oscillator.
- LSECSSON must be enabled after LSE is enabled (LSEON enabled) and ready (LSERDY set by hardware) and after RTCSEL is selected.
- Once enabled, this bit can only be disabled,
  - After a LSE failure detection (LSECSSD = 1). In that case the software must disable LSECSSON.
  - After a back-up domain reset
  - 0: CSS on 32 kHz oscillator OFF (default after Backup domain reset)
  - 1: CSS on 32 kHz oscillator ON

Bits 4:3 LSEDRV[1:0]: LSE oscillator driving capability
- Set by software to select the driving capability of the LSE oscillator.
  - 00: lowest drive
  - 01: medium-low drive
  - 10: medium-high drive (default after backup domain if LSEON=0)
  - 11: highest drive
### 7.8.32 RCC clock control and status register (RCC_CSR)

Address offset: 0x074

Reset value: 0x0000 0000

Access: 0 ≤ wait state ≤ 7, word, half-word and byte access

Wait states are inserted in case of successive accesses to this register.

<table>
<thead>
<tr>
<th>Bit 31:2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>LSIRDY</td>
<td>LSION</td>
</tr>
</tbody>
</table>

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

**Bit 1** **LSIRDY**: LSI oscillator ready
- Set and reset by hardware to indicate when the low-speed internal RC oscillator is stable.
- This bit needs 3 cycles of lsi_ck clock to fall down after LSION has been set to 0.
- This bit can be set even when LSION is not enabled if there is a request for LSI clock by the clock security system on LSE or by the low-speed watchdog or by the RTC.
- 0: LSI clock is not ready (default after reset)
- 1: LSI clock is ready

**Bit 0** **LSION**: LSI oscillator enable
- Set and reset by software.
- 0: LSI is OFF (default after reset)
- 1: LSI is ON
### 7.8.33 RCC AHB1 peripheral reset register (RCC_AHB1RSTR)

Address offset: 0x080  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>ADF1RST: ADF block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset ADF block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets ADF block</td>
</tr>
</tbody>
</table>

Bits 30:28 Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>OTGFSRST: OTGFS block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset OTGFS block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets OTGFS block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>USBPHYCRST: USBPHYC block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset USBPHYC block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets USBPHYC block</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>OTGHSRST: OTGHS block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset OTGHS block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets OTGHS block</td>
</tr>
</tbody>
</table>

Bits 24:16 Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>ETH1RST: ETH1 block reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset ETH1 block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets ETH1 block</td>
</tr>
</tbody>
</table>

Bits 14:6 Reserved, must be kept at reset value.
Bit 5 **ADC12RST**: ADC1 and 2 blocks reset
   Set and reset by software.
   0: does not reset ADC1 and 2 blocks (default after reset)
   1: resets ADC1 and 2 blocks

Bit 4 **GPDMA1RST**: GPDMA1 blocks reset
   Set and reset by software.
   0: does not reset GPDMA1 block (default after reset)
   1: resets GPDMA1 block

Bits 3:0 Reserved, must be kept at reset value.

### 7.8.34 **RCC AHB2 peripheral reset register (RCC_AHB2RSTR)**

Address offset: 0x084
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CORDICRST**: CORDIC reset
   Set and reset by software.
   0: does not reset CORDIC block (default after reset)
   1: resets CORDIC block

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SDMMC2RST**: SDMMC2 and SDMMC2 delay blocks reset
   Set and reset by software.
   0: does not reset SDMMC2 and SDMMC2 delay blocks (default after reset)
   1: resets SDMMC2 and SDMMC2 delay blocks

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **PSSIRST**: PSSI block reset
   Set and reset by software.
   0: does not reset PSSI block (default after reset)
   1: resets PSSI block

Bit 0 Reserved, must be kept at reset value.
7.8.35 RCC AHB3 peripheral reset register (RCC_AHB3RSTR)

Address offset: 0x0A4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PKARST**: PKA block reset
Set and reset by software.
0: does not reset PKA block (default after reset)
1: resets PKA block

Bit 5 Reserved, must be kept at reset value.

Bit 4 **SAESRST**: SAES block reset
Set and reset by software.
0: does not reset SAES block (default after reset)
1: resets SAES block

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPRST**: CRYP block reset
Set and reset by software.
0: does not reset CRYP block (default after reset)
1: resets CRYP block

Bit 1 **HASHRST**: HASH block reset
Set and reset by software.
0: does not reset HASH block (default after reset)
1: resets HASH block

Bit 0 **RNGRST**: random number generator block reset
Set and reset by software.
0: does not reset RNG block (default after reset)
1: resets RNG block
### 7.8.36  RCC AHB4 peripheral reset register (RCC_AHB4RSTR)

Address offset: 0x088  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:20</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 19</td>
<td><strong>CRCRST</strong>: CRC block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the CRC block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the CRC block</td>
</tr>
<tr>
<td>Bit 18:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 15</td>
<td><strong>GPIOPRST</strong>: GPIOP block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the GPIOP block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the GPIOP block</td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>GPIOORST</strong>: GPIOO block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the GPIOO block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the GPIOO block</td>
</tr>
<tr>
<td>Bit 13</td>
<td><strong>GPIONRST</strong>: GPION block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the GPION block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the GPION block</td>
</tr>
<tr>
<td>Bit 12</td>
<td><strong>GPIOMRST</strong>: GPIOM block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the GPIOM block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the GPIOM block</td>
</tr>
<tr>
<td>Bits 11:8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 7</td>
<td><strong>GPIOHRST</strong>: GPIOH block reset</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: does not reset the GPIOH block (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: resets the GPIOH block</td>
</tr>
</tbody>
</table>
Bit 6  **GPIOGRST**: GPIOG block reset
       Set and reset by software.
       0: does not reset the GPIOG block (default after reset)
       1: resets the GPIOG block

Bit 5  **GPIOFRST**: GPIOF block reset
       Set and reset by software.
       0: does not reset the GPIOF block (default after reset)
       1: resets the GPIOF block

Bit 4  **GPIOERST**: GPIOE block reset
       Set and reset by software.
       0: does not reset the GPIOE block (default after reset)
       1: resets the GPIOE block

Bit 3  **GPIODRST**: GPIOD block reset
       Set and reset by software.
       0: does not reset the GPIOD block (default after reset)
       1: resets the GPIOD block

Bit 2  **GPIOCRST**: GPIOC block reset
       Set and reset by software.
       0: does not reset the GPIOC block (default after reset)
       1: resets the GPIOC block

Bit 1  **GPIOBRST**: GPIOB block reset
       Set and reset by software.
       0: does not reset the GPIOB block (default after reset)
       1: resets the GPIOB block

Bit 0  **GPIOARST**: GPIOA block reset
       Set and reset by software.
       0: does not reset the GPIOA block (default after reset)
       1: resets the GPIOA block
7.8.37  RCC AHB5 peripheral reset register (RCC_AHB5RSTR)

Address offset: 0x07C
Reset value: 0x0000 0000

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20  **GPU2DRST**: GPU2D block reset
Set and reset by software.
0: reset is released (default after reset)
1: reset is asserted

Bit 19  **GFXMMURST**: GFXMMU block reset
Set and reset by software.
0: reset is released (default after reset)
1: reset is asserted

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

Bit 14  **XSPIMRST**: XSPIM reset
Set and reset by software.
0: reset is released (default after reset)
1: reset is asserted

Bit 13 Reserved, must be kept at reset value.

Bit 12  **XSPI2RST**: XSPI2 and MCE2 blocks reset
Set and reset by software.
The hardware prevents writing this bit if XSPICKP = 1.
0: reset is released (default after reset)
1: reset is asserted

Bits 11:9 Reserved, must be kept at reset value.

Bit 8  **SDMMC1RST**: SDMMC1 and DB_SDMMC1 blocks reset
Set and reset by software.
0: reset is released (default after reset)
1: reset is asserted

Bits 7:6 Reserved, must be kept at reset value.
Bit 5 **XSPI1RST**: XSPI1 and MCE1 blocks reset
   Set and reset by software.
   The hardware prevents writing this bit if XSPICKP = 1.
   0: reset is released (default after reset)
   1: reset is asserted

Bit 4 **FMCRST**: FMC and MCE3 blocks reset
   Set and reset by software.
   The hardware prevents writing this bit if FMCCKP = 1.
   0: reset is released (default after reset)
   1: reset is asserted

Bit 3 **JPEGRST**: JPEG block reset
   Set and reset by software.
   0: reset is released (default after reset)
   1: reset is asserted

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DRST**: DMA2D block reset
   Set and reset by software.
   0: reset is released (default after reset)
   1: reset is asserted

Bit 0 **HPDMA1RST**: HPDMA1 block reset
   Set and reset by software.
   0: reset is released (default after reset)
   1: reset is asserted
### 7.8.38 RCC APB1 peripheral reset register 1 (RCC_APB1RSTR1)

Address offset: 0x090
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Reset Value</th>
<th>Access</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UART8RST: UART8 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>UART7RST: UART7 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>CECRST: HDMI-CEC block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>I2C3RST: I2C3 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>I2C2RST: I2C2 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>I2C1_I3C1RST: I2C1/I3C1 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>UART6RST: UART6 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>UART5RST: UART5 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>UART4RST: UART4 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>UART3RST: UART3 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>UART2RST: UART2 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>UART1RST: UART1 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>SPI4RST: SPI4 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>SPI3RST: SPI3 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>SPI2RST: SPI2 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>SPI1RST: SPI1 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>TIM14RST: TIM14 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>TIM13RST: TIM13 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>TIM12RST: TIM12 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>TIM11RST: TIM11 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>TIM10RST: TIM10 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>TIM9RST: TIM9 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>TIM8RST: TIM8 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>TIM7RST: TIM7 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>TIM6RST: TIM6 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>TIM5RST: TIM5 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>TIM4RST: TIM4 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>TIM3RST: TIM3 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>TIM2RST: TIM2 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>TIM1RST: TIM1 block reset</td>
<td></td>
<td>rw</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bit 31** UART8RST: UART8 block reset
  - Set and reset by software.
  - 0: does not reset the UART8 block (default after reset)
  - 1: resets the UART8 block

- **Bit 30** UART7RST: UART7 block reset
  - Set and reset by software.
  - 0: does not reset the UART7 block (default after reset)
  - 1: resets the UART7 block

- **Bits 29:28** Reserved, must be kept at reset value.

- **Bit 27** CECRST: HDMI-CEC block reset
  - Set and reset by software.
  - 0: does not reset the HDMI-CEC block (default after reset)
  - 1: resets the HDMI-CEC block

- **Bits 26:24** Reserved, must be kept at reset value.

- **Bit 23** I2C3RST: I2C3 block reset
  - Set and reset by software.
  - 0: does not reset the I2C3 block (default after reset)
  - 1: resets the I2C3 block

- **Bit 22** I2C2RST: I2C2 block reset
  - Set and reset by software.
  - 0: does not reset the I2C2 block (default after reset)
  - 1: resets the I2C2 block

- **Bit 21** I2C1_I3C1RST: I2C1/I3C1 block reset
  - Set and reset by software.
  - 0: does not reset the I2C1/I3C1 block (default after reset)
  - 1: resets the I2C1/I3C1 block
Bit 20 **UART5RST**: UART5 block reset
Set and reset by software.
0: does not reset the UART5 block (default after reset)
1: resets the UART5 block

Bit 19 **UART4RST**: UART4 block reset
Set and reset by software.
0: does not reset the UART4 block (default after reset)
1: resets the UART4 block

Bit 18 **USART3RST**: USART3 block reset
Set and reset by software.
0: does not reset the USART3 block (default after reset)
1: resets the USART3 block

Bit 17 **USART2RST**: USART2 block reset
Set and reset by software.
0: does not reset the USART2 block (default after reset)
1: resets the USART2 block

Bit 16 **SPDIFRXRST**: SPDIFRX block reset
Set and reset by software.
0: does not reset the SPDIFRX block (default after reset)
1: resets the SPDIFRX block

Bit 15 **SPI3RST**: SPI2S3 block reset
Set and reset by software.
0: does not reset the SPI2S3 block (default after reset)
1: resets the SPI2S3 block

Bit 14 **SPI2RST**: SPI2S2 block reset
Set and reset by software.
0: does not reset the SPI2S2 block (default after reset)
1: resets the SPI2S2 block

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 block reset
Set and reset by software.
0: does not reset the LPTIM1 block (default after reset)
1: resets the LPTIM1 block

Bit 8 **TIM14RST**: TIM14 block reset
Set and reset by software.
0: does not reset the TIM14 block (default after reset)
1: resets the TIM14 block

Bit 7 **TIM13RST**: TIM13 block reset
Set and reset by software.
0: does not reset the TIM13 block (default after reset)
1: resets the TIM13 block

Bit 6 **TIM12RST**: TIM12 block reset
Set and reset by software.
0: does not reset the TIM12 block (default after reset)
1: resets the TIM12 block
Bit 5 **TIM7RST**: TIM7 block reset
Set and reset by software.
0: does not reset the TIM7 block (default after reset)
1: resets the TIM7 block

Bit 4 **TIM6RST**: TIM6 block reset
Set and reset by software.
0: does not reset the TIM6 block (default after reset)
1: resets the TIM6 block

Bit 3 **TIM5RST**: TIM5 block reset
Set and reset by software.
0: does not reset the TIM5 block (default after reset)
1: resets the TIM5 block

Bit 2 **TIM4RST**: TIM4 block reset
Set and reset by software.
0: does not reset the TIM4 block (default after reset)
1: resets the TIM4 block

Bit 1 **TIM3RST**: TIM3 block reset
Set and reset by software.
0: does not reset the TIM3 block (default after reset)
1: resets the TIM3 block

Bit 0 **TIM2RST**: TIM2 block reset
Set and reset by software.
0: does not reset the TIM2 block (default after reset)
1: resets the TIM2 block
## 7.8.39 RCC APB1 peripheral reset register 2 (RCC_APB1RSTR2)

Address offset: 0x094  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **UCPD1RST**: UCPD block reset  
Set and reset by software.  
0: does not reset the UCPD block (default after reset)  
1: resets the UCPD block

Bits 26:9 Reserved, must be kept at reset value.

Bit 8 **FDCANRST**: FDCAN block reset  
Set and reset by software.  
0: does not reset the FDCAN block (default after reset)  
1: resets the FDCAN block

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSRST**: MDIOS block reset  
Set and reset by software.  
0: does not reset the MDIOS block (default after reset)  
1: resets the MDIOS block

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CRSRST**: clock recovery system reset  
Set and reset by software.  
0: does not reset CRS (default after reset)  
1: resets CRS

Bit 0 Reserved, must be kept at reset value.
### 7.8.40  RCC APB2 peripheral reset register (RCC_APB2RSTR)

*Address offset: 0x098*

*Reset value: 0x0000 0000*

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----------</td>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----------</td>
<td>----------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:24**  Reserved, must be kept at reset value.

**Bit 23**  **SAI2RST**: SAI2 block reset  
Set and reset by software.  
0: does not reset the SAI2 (default after reset)  
1: resets the SAI2

**Bit 22**  **SAI1RST**: SAI1 block reset  
Set and reset by software.  
0: does not reset the SAI1 (default after reset)  
1: resets the SAI1

**Bit 21**  Reserved, must be kept at reset value.

**Bit 20**  **SPI5RST**: SPI5 block reset  
Set and reset by software.  
0: does not reset the SPI5 block (default after reset)  
1: resets the SPI5 block

**Bit 19**  **TIM9RST**: TIM9 block reset  
Set and reset by software.  
0: does not reset the TIM9 block (default after reset)  
1: resets the TIM9 block

**Bit 18**  **TIM17RST**: TIM17 block reset  
Set and reset by software.  
0: does not reset the TIM17 block (default after reset)  
1: resets the TIM17 block

**Bit 17**  **TIM16RST**: TIM16 block reset  
Set and reset by software.  
0: does not reset the TIM16 block (default after reset)  
1: resets the TIM16 block

**Bit 16**  **TIM15RST**: TIM15 block reset  
Set and reset by software.  
0: does not reset the TIM15 block (default after reset)  
1: resets the TIM15 block
Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SPI4RST**: SPI4 block reset
Set and reset by software.
0: does not reset the SPI4 block (default after reset)
1: resets the SPI4 block

Bit 12 **SPI1RST**: SPI2S1 block reset
Set and reset by software.
0: does not reset the SPI2S1 block (default after reset)
1: resets the SPI2S1 block

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **USART1RST**: USART1 block reset
Set and reset by software.
0: does not reset the USART1 block (default after reset)
1: resets the USART1 block

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TIM1RST**: TIM1 block reset
Set and reset by software.
0: does not reset the TIM1 block (default after reset)
1: resets the TIM1 block
7.8.41 RCC APB4 peripheral reset register (RCC_APB4RSTR)

Address offset: 0x09C
Reset value: 0x0000 0000

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

Bit 26 DTSRST: DTS block reset
Set and reset by software.
0: does not reset the DTS block (default after reset)
1: resets the DTS block

Bits 25:16 Reserved, must be kept at reset value.

Bit 15 VREFRST: VREF block reset
Set and reset by software.
0: does not reset the VREF block (default after reset)
1: resets the VREF block

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 LPTIM5RST: LPTIM5 block reset
Set and reset by software.
0: does not reset the LPTIM5 block (default after reset)
1: resets the LPTIM5 block

Bit 11 LPTIM4RST: LPTIM4 block reset
Set and reset by software.
0: does not reset the LPTIM4 block (default after reset)
1: resets the LPTIM4 block

Bit 10 LPTIM3RST: LPTIM3 block reset
Set and reset by software.
0: does not reset the LPTIM3 block (default after reset)
1: resets the LPTIM3 block

Bit 9 LPTIM2RST: LPTIM2 block reset
Set and reset by software.
0: does not reset the LPTIM2 block (default after reset)
1: resets the LPTIM2 block

Bits 8:6 Reserved, must be kept at reset value.
Bit 5  **SPI6RST**: SPI/I2S6 block reset
   Set and reset by software.
   0: does not reset the SPI/I2S6 block (default after reset)
   1: resets the SPI/I2S6 block

Bit 4  Reserved, must be kept at reset value.

Bit 3  **LPUART1RST**: LPUART1 block reset
   Set and reset by software.
   0: does not reset the LPUART1 block (default after reset)
   1: resets the LPUART1 block

Bit 2  Reserved, must be kept at reset value.

Bit 1  **SBSRST**: SBS block reset
   Set and reset by software.
   0: does not reset the SBS block (default after reset)
   1: resets the SBS block

Bit 0  Reserved, must be kept at reset value.
### 7.8.42 RCC APB5 peripheral reset register (RCC_APB5RSTR)

**Address offset:** 0x08C  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-5</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 4 | GFTXIMRST | GFXTIM block reset  
Set and reset by software.  
0: does not reset the GFXTIM block (default after reset)  
1: resets the GFXTIM block |
| 3 | Reserved | Must be kept at reset value. |
| 2 | DCMIPPRST | DCMIPP block reset  
Set and reset by software.  
0: does not reset the DCMIPP block (default after reset)  
1: resets the DCMIPP block |
| 1 | LTDCRST | LTDC block reset  
Set and reset by software.  
0: does not reset the LTDC block (default after reset)  
1: resets the LTDC block |
| 0 | Reserved | Must be kept at reset value. |
### 7.8.43 RCC AXI clocks gating disable register (RCC_CKGDISR)

Address offset: 0x0B0  
Reset value: 0x8000 0000

The dynamic power consumption can be optimized by enabling the functional clock gating.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Value at Reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>JTAGCKG</td>
<td>JTAG automatic clock gating disabling</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>EXTICKG</td>
<td>EXTI clock gating disable</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

**Bit 31 JTAGCKG**: JTAG automatic clock gating disabling  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock is disabled except if a JTAG connection has been detected  
1: The clock gating is disabled. The clock is always enabled. (default after reset)

**Bit 30 EXTICKG**: EXTI clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock is enabled after an event detection and stopped again when the event flag is cleared. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

**Bits 29:22 Reserved**, must be kept at reset value.

**Bit 21 FLASHCKG**: AXI slave Flash interface (FLIFT) clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI slave port connected to the FLASH is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

**Bit 20 AXISRAM1CKG**: AXI slave SRAM1 / error code correction (ECC) clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI slave port connected to the SRAM1 is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

**Bit 19 AXISRAM2CKG**: AXI slave SRAM2 clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI slave port connected to the SRAM2 is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.
Bit 18 **AXISRAM3CKG**: AXI matrix slave SRAM3 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI slave port connected to the SRAM3 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 17 **AXISRAM4CKG**: AXI matrix slave SRAM4 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI slave port connected to the SRAM4 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 16 **XSPI2CKG**: AXI slave XSPI2 and MCE2 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI slave port connected to the XSPI2 and MCE2 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 15 **XSPI1CKG**: AXI slave XSPI1 and MCE1 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI slave port connected to the XSPI1 and MCE1 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 14 **FMCCKG**: AXI slave FMC and MCE3 clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI slave port connected to the FMC and MCE3 is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 13 **AHBSCKG**: AXI slave AHB clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The AXI matrix slave AHB clock is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 12 **GFXMMUMCKG**: AXI master GFXMMU clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the GFXMMU is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 11 **LTDCCKG**: AXI master LTDC clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI master port connected to the LTDC is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled.

Bit 10 **GFXMMUSCKG**: AXI matrix slave GFXMMU clock gating disable
This bit is set and reset by software.
0: The clock gating is enabled. The clock of the AXI slave port connected to the GFXMMU is enabled on bus transaction request. (default after reset)
1: The clock gating is disabled. The clock is always enabled
Bit 9 **DMA2DCKG**: AXI master DMA2D clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the DMA2D is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled

Bit 8 **DCMIPPCKG**: AXI master DCMIPP clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the DCMIPP is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

Bit 7 **GPU2DCLCKG**: AXI master cache GPU2D clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the GPU2D I-Cache is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

Bit 6 **GPU2D$S1CKG**: AXI master 1 GPU2D clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the GPU2D port 1 is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

Bit 5 **GPU2D$S0CKG**: AXI master 0 GPU2D clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the GPU2D port 0 is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

Bit 4 **CPUCKG**: AXI master CPU clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the CPU is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.

Bit 3 **HPDMA1CKG**: AXI master HPDMA1 clock gating disable  
This bit is set and reset by software.  
0: The clock gating is enabled. The clock of the AXI master port connected to the HPDMA1 is enabled on bus transaction request. (default after reset)  
1: The clock gating is disabled. The clock is always enabled.
Bit 2  **SDMMC1CKG**: AXI master SDMMC1 clock gating disable
   This bit is set and reset by software.
   0: The clock gating is enabled. The clock of the AXI master port connected to SDMMC1 is enabled on bus transaction request. (default after reset)
   1: The clock gating is disabled. The clock is always enabled.

Bit 1  **AHBMCKG**: AXI master AHB clock gating disable
   This bit is set and reset by software.
   0: The clock gating is enabled. The clock of the AXI master port connected to the AHB interconnect is enabled on bus transaction request. (default after reset)
   1: The clock gating is disabled. The clock is always enabled.

Bit 0  **AXICKG**: AXI interconnect matrix clock gating disable
   This bit is set and reset by software.
   0: The clock gating is enabled. The AXI interconnect matrix clock is enabled on bus transaction request. (default after reset)
   1: The clock gating is disabled. The clock is always enabled
## 7.8.44 RCC Reset status register (RCC_RSR)

Address offset: 0x130  
Reset value: 0x00E0 0000  
Reset by power-on reset only.

Access: $0 \leq$ wait state $\leq 7$, word, half-word and byte access. Wait states are inserted in case of successive accesses to this register.

### Bit Descriptions

#### Bit 31 Reserve, must be kept at reset value.

#### Bit 30 LPWRRSTF: reset due to illegal Stop or Standby flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when the CPU goes erroneously in Stop or Standby mode,
  - 0: no illegal reset occurred (default after power-on reset)
  - 1: illegal Stop or Standby reset occurred

#### Bit 29 Reserve, must be kept at reset value.

#### Bit 28 WWDGRSTF: window watchdog reset flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when a window watchdog reset occurs.
  - 0: no window watchdog reset occurred from WWDG (default after power-on reset)
  - 1: window watchdog reset occurred from WWDG

#### Bit 27 Reserve, must be kept at reset value.

#### Bit 26 IWGRSTF: independent watchdog reset flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when an independent watchdog reset occurs.
  - 0: no independent watchdog reset occurred (default after power-on reset)
  - 1: independent watchdog reset occurred

#### Bit 25 Reserve, must be kept at reset value.

#### Bit 24 SFTRSTF: system reset from CPU reset flag \(^{(1)}\)
- Reset by software by writing the RMVF bit.
- Set by hardware when the system reset is due to CPU. The CPU can generate a system reset by writing SYSRESETREQ bit of AIRCR register of the core M7.
  - 0: no CPU software reset occurred (default after power-on reset)
  - 1: a system reset has been generated by the CPU

### Register Layout

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserve</td>
</tr>
<tr>
<td>30</td>
<td>LPWRRSTF</td>
</tr>
<tr>
<td>29</td>
<td>Reserve</td>
</tr>
<tr>
<td>28</td>
<td>WWDGRSTF</td>
</tr>
<tr>
<td>27</td>
<td>Reserve</td>
</tr>
<tr>
<td>26</td>
<td>IWGRSTF</td>
</tr>
<tr>
<td>25</td>
<td>Reserve</td>
</tr>
<tr>
<td>24</td>
<td>SFTRSTF</td>
</tr>
</tbody>
</table>
Bit 23 **PORRSTF**: POR/PDR reset flag (1)
- Reset by software by writing the RMVF bit.
- Set by hardware when a POR/PDR reset occurs.
  - 0: no POR/PDR reset occurred
  - 1: POR/PDR reset occurred (default after power-on reset)

Bit 22 **PINRSTF**: pin reset flag (NRST) (1)
- Reset by software by writing the RMVF bit.
- Set by hardware when a reset from pin occurs.
  - 0: no reset from pin occurred
  - 1: reset from pin occurred (default after power-on reset)

Bit 21 **BORRSTF**: BOR reset flag (1)
- Reset by software by writing the RMVF bit.
- Set by hardware when a BOR reset occurs (pwr_bor_rst).
  - 0: no BOR reset occurred
  - 1: BOR reset occurred (default after power-on reset)

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **OBLRSTF**: Option byte loading reset flag (1)
- Reset by software by the RMVF bit.
- Set by hardware when a reset from the option byte loading occurs.
  - 0: No reset from option byte loading occurred
  - 1: Reset from option byte loading occurred

Bit 16 **RMVF**: remove reset flag
- Set and reset by software to reset the value of the reset flags.
  - 0: reset of the reset flags not activated (default after power-on reset)
  - 1: resets the value of the reset flags

Bits 15:0 Reserved, must be kept at reset value.

1. Refer to *Table 57: Reset source identification (RCC_RSR)* for details on flag behavior.
7.8.45  **RCC AHB1 clock enable register (RCC_AHB1ENR)**

Address offset: 0x138
Reset value: 0x0000 0000

| Bit 31 | ADF1EN: ADF clocks enable
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: ADF clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: ADF clocks enabled</td>
</tr>
</tbody>
</table>

| Bits 30:28 | Reserved, must be kept at reset value.                      |

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>OTGFSEN: OTGFS peripheral clocks enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: OTGFS peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: OTGFS peripheral clocks enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>USBPHYCEN: USBPHYC clocks enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: USBPHYC clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: USBPHYC clocks enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>OTGHSEN: OTGHS clocks enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: OTGHS clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: OTGHS clocks enabled</td>
</tr>
</tbody>
</table>

| Bits 24:18 | Reserved, must be kept at reset value.                     |

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>ETH1RXEN: ETH1 reception clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: ETH1 reception clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: ETH1 reception clock enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>ETH1TXEN: ETH1 transmission clock enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: ETH1 transmission clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: ETH1 transmission clock enabled</td>
</tr>
</tbody>
</table>
Bit 15 **ETH1MACEN**: ETH1 MAC peripheral clock enable
  Set and reset by software.
  0: ETH1 MAC peripheral clock disabled (default after reset)
  1: ETH1 MAC peripheral clock enabled

Bits 14:6 Reserved, must be kept at reset value.

Bit 5 **ADC12EN**: ADC1 and 2 peripheral clocks enable
  Set and reset by software.
  0: ADC1 and 2 peripheral clocks disabled (default after reset)
  1: ADC1 and 2 peripheral clocks enabled
  The peripheral clocks of the ADC1 and 2 are the kernel clock selected by ADCSEL and provided to
  ADCx_CK input, and the **hclk1** bus interface clock.

Bit 4 **GPDMA1EN**: GPDMA1 clock enable
  Set and reset by software.
  0: GPDMA1 clock disabled (default after reset)
  1: GPDMA1 clock enabled

Bits 3:0 Reserved, must be kept at reset value.
7.8.46 RCC AHB2 clock enable register (RCC_AHB2ENR)

Address offset: 0x13C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>reset value</th>
<th>default after reset</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>SRAM2EN: SRAM2 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>SRAM1EN: SRAM1 clock enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SRAM2EN**: SRAM2 clock enable
Set and reset by software.
0: SRAM2 clock disabled (default after reset)
1: SRAM2 clock enabled

Bit 29 **SRAM1EN**: SRAM1 clock enable
Set and reset by software.
0: SRAM1 clock disabled (default after reset)
1: SRAM1 clock enabled

Bits 28:15 Reserved, must be kept at reset value.

Bit 14 **CORDICEN**: CORDIC clock enable
Set and reset by software.
0: CORDIC clock disabled (default after reset)
1: CORDIC clock enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **SDMMC2EN**: SDMMC2 and SDMMC2 delay clock enable
Set and reset by software.
0: SDMMC2 and SDMMC2 delay clock disabled (default after reset)
1: SDMMC2 and SDMMC2 delay clock enabled

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **PSSIEN**: PSSI peripheral clocks enable
Set and reset by software.
0: PSSI peripheral clocks disabled (default after reset)
1: PSSI peripheral clocks enabled

Bit 0 Reserved, must be kept at reset value.
### 7.8.47 RCC AHB3 clock enable register (RCC_AHB3ENR)

Address offset: 0x158
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

**Bit 6 PKAEN**: PKA peripheral clock enable
- Set and reset by software.
  - 0: PKA peripheral clock disabled (default after reset)
  - 1: PKA peripheral clock enabled

**Bit 5** Reserved, must be kept at reset value.

**Bit 4 SAESEN**: SAES peripheral clock enable
- Set and reset by software.
  - This bit controls the enable of the clock delivered to the SAES.
  - 0: The SAES peripheral clocks are disabled (default after reset)
  - 1: The SAES peripheral clocks are enabled

**Bit 3** Reserved, must be kept at reset value.

**Bit 2 CRYPEN**: CRYP peripheral clock enable
- Set and reset by software.
  - 0: CRYP peripheral clock disabled (default after reset)
  - 1: CRYP peripheral clock enabled

**Bit 1 HASHEN**: HASH peripheral clock enable
- Set and reset by software.
  - 0: HASH peripheral clock disabled (default after reset)
  - 1: HASH peripheral clock enabled

**Bit 0 RNGEN**: RNG peripheral clocks enable
- Set and reset by software.
  - 0: RNG peripheral clocks disabled (default after reset)
  - 1: RNG peripheral clocks enabled.
7.8.48  RCC AHB4 clock enable register (RCC_AHB4ENR)

Address offset: 0x140  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:29  Reserved, must be kept at reset value.

Bit 28  BKPRA: Backup RAM clock enable
Set and reset by software.
0: Backup RAM clock disabled (default after reset)
1: Backup RAM clock enabled

Bits 27:20  Reserved, must be kept at reset value.

Bit 19  CRCEN: CRC clock enable
Set and reset by software.
0: CRC clock disabled (default after reset)
1: CRC clock enabled

Bits 18:16  Reserved, must be kept at reset value.

Bit 15  GPIOPEN: GPIOP peripheral clock enable
Set and reset by software.
0: GPIOP peripheral clock disabled (default after reset)
1: GPIOP peripheral clock enabled

Bit 14  GPIOOEN: GPIOO peripheral clock enable
Set and reset by software.
0: GPIOO peripheral clock disabled (default after reset)
1: GPIOO peripheral clock enabled

Bit 13  GPIONEN: GPION peripheral clock enable
Set and reset by software.
0: GPION peripheral clock disabled (default after reset)
1: GPION peripheral clock enabled

Bit 12  GPIOMEN: GPIOM peripheral clock enable
Set and reset by software.
0: GPIOM peripheral clock disabled (default after reset)
1: GPIOM peripheral clock enabled

Bits 11:8  Reserved, must be kept at reset value.
Bit 7  **GPIOHEN**: GPIOH peripheral clock enable
Set and reset by software.
0: GPIOH peripheral clock disabled (default after reset)
1: GPIOH peripheral clock enabled

Bit 6  **GPIOGEN**: GPIOG peripheral clock enable
Set and reset by software.
0: GPIOG peripheral clock disabled (default after reset)
1: GPIOG peripheral clock enabled

Bit 5  **GPIOFEN**: GPIOF peripheral clock enable
Set and reset by software.
0: GPIOF peripheral clock disabled (default after reset)
1: GPIOF peripheral clock enabled

Bit 4  **GPIOEEN**: GPIOE peripheral clock enable
Set and reset by software.
0: GPIOE peripheral clock disabled (default after reset)
1: GPIOE peripheral clock enabled

Bit 3  **GPIODEN**: GPIOD peripheral clock enable
Set and reset by software.
0: GPIOD peripheral clock disabled (default after reset)
1: GPIOD peripheral clock enabled

Bit 2  **GPIOCEN**: GPIOC peripheral clock enable
Set and reset by software.
0: GPIOC peripheral clock disabled (default after reset)
1: GPIOC peripheral clock enabled

Bit 1  **GPIOBEN**: GPIOB peripheral clock enable
Set and reset by software.
0: GPIOB peripheral clock disabled (default after reset)
1: GPIOB peripheral clock enabled

Bit 0  **GPIOAEN**: GPIOA peripheral clock enable
Set and reset by software.
0: GPIOA peripheral clock disabled (default after reset)
1: GPIOA peripheral clock enabled
### 7.8.49 RCC AHB5 clock enable register (RCC_AHB5ENR)

**Address offset:** 0x134  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

**Bit 20 GPU2DEN**: GPU2D peripheral clock enable  
Set and reset by software.  
0: GPU2D peripheral clock disabled (default after reset)  
1: GPU2D peripheral clock enabled

**Bit 19 GFXMMUEN**: GFXMMU peripheral clock enable  
Set and reset by software.  
0: GFXMMU peripheral clock disabled (default after reset)  
1: GFXMMU peripheral clock enabled

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

**Bit 14 XSPIMEN**: XSPIM peripheral clock enable  
Set and reset by software.  
0: XSPIM peripheral clock disabled (default after reset)  
1: XSPIM peripheral clock enabled

**Bit 13 Reserved, must be kept at reset value.**

**Bit 12 XSPI2EN**: XSPI2 and MCE2 peripheral clocks enable  
Set and reset by software.  
The hardware prevents writing this bit if XSPICKP = 1.  
0: XSPI2 and MCE2 peripheral clocks disabled (default after reset)  
1: XSPI2 and MCE2 peripheral clocks enabled

Bits 11:9 Reserved, must be kept at reset value.

**Bit 8 SDMMC1EN**: SDMMC1 and DB_SDMMC1 peripheral clocks enable  
Set and reset by software.  
0: SDMMC1 and DB_SDMMC1 peripheral clocks disabled (default after reset)  
1: SDMMC1 and DB_SDMMC1 peripheral clocks enabled

Bits 7:6 Reserved, must be kept at reset value.
Bit 5 **XSPI1EN**: XSPI1 and MCE1 peripheral clocks enable
- Set and reset by software.
  - The hardware prevents writing this bit if XSPICKP = 1.
  - 0: XSPI1 and MCE1 peripheral clocks disabled (default after reset)
  - 1: XSPI1 and MCE1 peripheral clocks enabled

Bit 4 **FMCEN**: FMC and MCE3 peripheral clocks enable
- Set and reset by software.
  - The hardware prevents writing this bit if FMCCKP = 1.
  - 0: FMC and MCE3 peripheral clocks disabled (default after reset)
  - 1: FMC and MCE3 peripheral clocks enabled
  - The peripheral clocks of the FMC are the kernel clock selected by FMCSEL, and the **hclk5** bus interface clock.

Bit 3 **JPEGEN**: JPEG peripheral clock enable
- Set and reset by software.
  - 0: JPEG peripheral clock disabled (default after reset)
  - 1: JPEG peripheral clock enabled

Bit 2 **Reserved**, must be kept at reset value.

Bit 1 **DMA2DEN**: DMA2D peripheral clock enable
- Set and reset by software.
  - 0: DMA2D peripheral clock disabled (default after reset)
  - 1: DMA2D peripheral clock enabled

Bit 0 **HPDMA1EN**: HPDMA1 peripheral clock enable
- Set and reset by software.
  - 0: HPDMA1 peripheral clock disabled (default after reset)
  - 1: HPDMA1 peripheral clock enabled
### 7.8.50 RCC APB1 clock enable register 1 (RCC_APB1ENR1)

**Address offset**: 0x148  
**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>UART8EN</th>
<th>UART7EN</th>
<th>UART6EN</th>
<th>UART5EN</th>
<th>UART4EN</th>
<th>UART3EN</th>
<th>UART2EN</th>
<th>UART1EN</th>
<th>UART0EN</th>
<th>SPI4EN</th>
<th>SPI3EN</th>
<th>SPI2EN</th>
<th>SPI1EN</th>
<th>WWDCEN</th>
<th>LPTIM1EN</th>
<th>TIM14EN</th>
<th>TIM13EN</th>
<th>TIM12EN</th>
<th>TIM11EN</th>
<th>TIM10EN</th>
<th>TIM9EN</th>
<th>TIM8EN</th>
<th>TIM7EN</th>
<th>TIM6EN</th>
<th>TIM5EN</th>
<th>TIM4EN</th>
<th>TIM3EN</th>
<th>TIM2EN</th>
<th>TIM1EN</th>
<th>TIM0EN</th>
<th>SPDIFRXEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31** **UART8EN**: UART8 peripheral clocks enable  
- Set and reset by software.  
- 0: UART8 peripheral clocks disabled (default after reset)  
- 1: UART8 peripheral clocks enabled

**Bit 30** **UART7EN**: UART7 peripheral clocks enable  
- Set and reset by software.  
- 0: UART7 peripheral clocks disabled (default after reset)  
- 1: UART7 peripheral clocks enabled

**Bits 29:28** **Reserved**, must be kept at reset value.

**Bit 27** **CECEN**: HDMI-CEC peripheral clock enable  
- Set and reset by software.  
- 0: HDMI-CEC peripheral clock disabled (default after reset)  
- 1: HDMI-CEC peripheral clock enabled

**Bits 26:24** **Reserved**, must be kept at reset value.

**Bit 23** **I2C3EN**: I2C3 peripheral clocks enable  
- Set and reset by software.  
- 0: I2C3 peripheral clocks disabled (default after reset)  
- 1: I2C3 peripheral clocks enabled

**Bit 22** **I2C2EN**: I2C2 peripheral clocks enable  
- Set and reset by software.  
- 0: I2C2 peripheral clocks disabled (default after reset)  
- 1: I2C2 peripheral clocks enabled

**Bit 21** **I2C1/I3C1EN**: I2C1/I3C1 peripheral clocks enable  
- Set and reset by software.  
- 0: I2C1/I3C1 peripheral clocks disabled (default after reset)  
- 1: I2C1/I3C1 peripheral clocks enabled

**Bit 20** **UART5EN**: UART5 peripheral clocks enable  
- Set and reset by software.  
- 0: UART5 peripheral clocks disabled (default after reset)  
- 1: UART5 peripheral clocks enabled
Bit 19 **UART4EN**: UART4 peripheral clocks enable
- Set and reset by software.
- 0: UART4 peripheral clocks disabled (default after reset)
- 1: UART4 peripheral clocks enabled
  The peripheral clocks of the UART4 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the \texttt{rcc\_pclk1} bus interface clock.

Bit 18 **USART3EN**: USART3 peripheral clocks enable
- Set and reset by software.
- 0: USART3 peripheral clocks disabled (default after reset)
- 1: USART3 peripheral clocks enabled
  The peripheral clocks of the USART3 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the \texttt{rcc\_pclk1} bus interface clock.

Bit 17 **USART2EN**: USART2 peripheral clocks enable
- Set and reset by software.
- 0: USART2 peripheral clocks disabled (default after reset)
- 1: USART2 peripheral clocks enabled
  The peripheral clocks of the USART2 are the kernel clock selected by USART234578SEL and provided to UCLK input, and the \texttt{rcc\_pclk1} bus interface clock.

Bit 16 **SPDIFRXEN**: SPDIFRX peripheral clocks enable
- Set and reset by software.
- 0: SPDIFRX peripheral clocks disabled (default after reset)
- 1: SPDIFRX peripheral clocks enabled
  The peripheral clocks of the SPDIFRX are the kernel clock selected by SPDIFRXSEL and provided to SPDIFRX\_CLK input, and the \texttt{rcc\_pclk1} bus interface clock.

Bit 15 **SPI3EN**: SPI3 peripheral clocks enable
- Set and reset by software.
- 0: SPI3 peripheral clocks disabled (default after reset)
- 1: SPI3 peripheral clocks enabled
  The peripheral clocks of the SPI3 are the kernel clock selected by I2S123SRC and provided to \texttt{com\_clk} input, and the \texttt{rcc\_pclk1} bus interface clock.

Bit 14 **SPI2EN**: SPI2 peripheral clocks enable
- Set and reset by software.
- 0: SPI2 peripheral clocks disabled (default after reset)
- 1: SPI2 peripheral clocks enabled
  The peripheral clocks of the SPI2 are the kernel clock selected by I2S123SRC and provided to \texttt{com\_clk} input, and the \texttt{rcc\_pclk1} bus interface clock.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGEN**: WWDG clock enable
- Set by software, and reset by hardware when a system reset occurs.
- 0: WWDG peripheral clock disable (default after reset)
- 1: WWDG peripheral clock enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 peripheral clocks enable
- Set and reset by software.
- 0: LPTIM1 peripheral clocks disabled (default after reset)
- 1: LPTIM1 peripheral clocks enabled
  The peripheral clocks of the LPTIM1 are the kernel clock selected by LPTIM1SEL and provided to \texttt{clk\_lpt} input, and the \texttt{rcc\_pclk1} bus interface clock.
Bit 8 **TIM14EN**: TIM14 peripheral clock enable
   Set and reset by software.
   0: TIM14 peripheral clock disabled (default after reset)
   1: TIM14 peripheral clock enabled

Bit 7 **TIM13EN**: TIM13 peripheral clock enable
   Set and reset by software.
   0: TIM13 peripheral clock disabled (default after reset)
   1: TIM13 peripheral clock enabled

Bit 6 **TIM12EN**: TIM12 peripheral clock enable
   Set and reset by software.
   0: TIM12 peripheral clock disabled (default after reset)
   1: TIM12 peripheral clock enabled

Bit 5 **TIM7EN**: TIM7 peripheral clock enable
   Set and reset by software.
   0: TIM7 peripheral clock disabled (default after reset)
   1: TIM7 peripheral clock enabled

Bit 4 **TIM6EN**: TIM6 peripheral clock enable
   Set and reset by software.
   0: TIM6 peripheral clock disabled (default after reset)
   1: TIM6 peripheral clock enabled

Bit 3 **TIM5EN**: TIM5 peripheral clock enable
   Set and reset by software.
   0: TIM5 peripheral clock disabled (default after reset)
   1: TIM5 peripheral clock enabled

Bit 2 **TIM4EN**: TIM4 peripheral clock enable
   Set and reset by software.
   0: TIM4 peripheral clock disabled (default after reset)
   1: TIM4 peripheral clock enabled

Bit 1 **TIM3EN**: TIM3 peripheral clock enable
   Set and reset by software.
   0: TIM3 peripheral clock disabled (default after reset)
   1: TIM3 peripheral clock enabled

Bit 0 **TIM2EN**: TIM2 peripheral clock enable
   Set and reset by software.
   0: TIM2 peripheral clock disabled (default after reset)
   1: TIM2 peripheral clock enabled
7.8.51 RCC APB1 clock enable register 2 (RCC_APB1ENR2)

Address offset: 0x14C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **UCPD1EN**: UCPD peripheral clock enable
Set and reset by software.
0: UCPD peripheral clock disabled (default after reset)
1: UCPD peripheral clock enabled

Bits 26:9 Reserved, must be kept at reset value.

Bit 8 **FDCANEN**: FDCAN peripheral clock enable
Set and reset by software.
0: FDCAN peripheral clock disabled (default after reset)
1: FDCAN peripheral clock enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSEN**: MDIOS peripheral clock enable
Set and reset by software.
0: MDIOS peripheral clock disabled (default after reset)
1: MDIOS peripheral clock enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CRSEN**: clock recovery system peripheral clock enable
Set and reset by software.
0: CRS peripheral clock disabled (default after reset)
1: CRS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.
7.8.52 RCC APB2 clock enable register (RCC_APB2ENR)

Address offset: 0x150
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 23</td>
<td><strong>SAI2EN</strong>: SAI2 peripheral clocks enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: SAI2 peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: SAI2 peripheral clocks enabled:</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the SAI2 are the kernel clock selected by SAI2SEL, and the pclk2 bus interface clock.</td>
</tr>
<tr>
<td>Bit 22</td>
<td><strong>SAI1EN</strong>: SAI1 peripheral clocks enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: SAI1 peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: SAI1 peripheral clocks enabled:</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the SAI1 are the kernel clock selected by SAI1SEL, and the pclk2 bus interface clock.</td>
</tr>
<tr>
<td>Bit 21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>SPI5EN</strong>: SPI5 peripheral clocks enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: SPI5 peripheral clocks disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: SPI5 peripheral clocks enabled:</td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the SPI5 are the kernel clock selected by SPI45SEL, and the pclk2 bus interface clock.</td>
</tr>
<tr>
<td>Bit 19</td>
<td><strong>TIM9EN</strong>: TIM9 peripheral clock enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: TIM9 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: TIM9 peripheral clock enabled</td>
</tr>
<tr>
<td>Bit 18</td>
<td><strong>TIM17EN</strong>: TIM17 peripheral clock enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: TIM17 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: TIM17 peripheral clock enabled</td>
</tr>
<tr>
<td>Bit 17</td>
<td><strong>TIM16EN</strong>: TIM16 peripheral clock enable</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: TIM16 peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td></td>
<td>1: TIM16 peripheral clock enabled</td>
</tr>
</tbody>
</table>
Bit 16  **TIM15EN**: TIM15 peripheral clock enable  
Set and reset by software.  
0: TIM15 peripheral clock disabled (default after reset)  
1: TIM15 peripheral clock enabled

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

Bit 13  **SPI4EN**: SPI4 Peripheral Clocks Enable  
Set and reset by software.  
0: SPI4 peripheral clocks disabled (default after reset)  
1: SPI4 peripheral clocks enabled:  
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI4SEL, and the `pclk2` bus interface clock.

Bit 12  **SPI1EN**: SPI2S1 Peripheral Clocks Enable  
Set and reset by software.  
0: SPI2S1 peripheral clocks disabled (default after reset)  
1: SPI2S1 peripheral clocks enabled:  
The peripheral clocks of the SPI2S1 are: the kernel clock selected by SPI1SEL, and the `pclk2` bus interface clock.

Bits 11:5  Reserved, must be kept at reset value.

Bit 4  **USART1EN**: USART1 peripheral clocks enable  
Set and reset by software.  
0: USART1 peripheral clocks disabled (default after reset)  
1: USART1 peripheral clocks enabled:  
The peripheral clocks of the USART1 are the kernel clock selected by USART1SEL, and the `pclk2` bus interface clock.

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **TIM1EN**: TIM1 peripheral clock enable  
Set and reset by software.  
0: TIM1 peripheral clock disabled (default after reset)  
1: TIM1 peripheral clock enabled

### 7.8.53  **RCC APB4 clock enable register (RCC_APB4ENR)**

Address offset: 0x154  
Reset value: 0x0001 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

---

[RM0477 Rev 6] 509/3791
Bits 31:27  Reserved, must be kept at reset value.

  Bit 26  **DTSEN**: Temperature Sensor peripheral clock enable
  Set and reset by software.
  0: DTS peripheral clock disabled (default after reset)
  1: DTS peripheral clock enabled

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

  Bit 16  **RTCAPBEN**: RTC APB clock enable
  Set and reset by software.
  0: The register clock interface of the RTC (APB) is disabled
  1: The register clock interface of the RTC (APB) is enabled (default after reset)

  Bit 15  **VREFEN**: VREF peripheral clock enable
  Set and reset by software.
  0: VREF peripheral clock disabled (default after reset)
  1: VREF peripheral clock enabled

Bits 14:13  Reserved, must be kept at reset value.

  Bit 12  **LPTIM5EN**: LPTIM5 peripheral clocks enable
  Set and reset by software.
  0: LPTIM5 peripheral clocks disabled (default after reset)
  1: LPTIM5 peripheral clocks enabled
  The LPTIM5 kernel clock can be selected by LPTIM45SEL.

  Bit 11  **LPTIM4EN**: LPTIM4 peripheral clocks enable
  Set and reset by software.
  0: LPTIM4 peripheral clocks disabled (default after reset)
  1: LPTIM4 peripheral clocks enabled
  The LPTIM4 kernel clock can be selected by LPTIM45SEL.

  Bit 10  **LPTIM3EN**: LPTIM3 peripheral clocks enable
  Set and reset by software.
  0: LPTIM3 peripheral clocks disabled (default after reset)
  1: LPTIM3 peripheral clocks enabled
  The LPTIM3 kernel clock can be selected by LPTIM23SEL.

  Bit 9  **LPTIM2EN**: LPTIM2 peripheral clocks enable
  Set and reset by software.
  0: LPTIM2 peripheral clocks disabled (default after reset)
  1: LPTIM2 peripheral clocks enabled
  The LPTIM2 kernel clock can be selected by LPTIM23SEL.

Bits 8:6  Reserved, must be kept at reset value.

  Bit 5  **SPI6EN**: SPI/I2S6 peripheral clocks enable
  Set and reset by software.
  0: SPI/I2S6 peripheral clocks disabled (default after reset)
  1: SPI/I2S6 peripheral clocks enabled
  The peripheral clocks of the SPI/I2S6 are the kernel clock selected by SPI6SEL and provided to `com_clk` input, and the `pclk4` bus interface clock.

  Bit 4  Reserved, must be kept at reset value.
Bit 3 **LPUART1EN**: LPUART1 peripheral clocks enable
   Set and reset by software.
   0: LPUART1 peripheral clocks disabled (default after reset)
   1: LPUART1 peripheral clocks enabled
   The peripheral clocks of the LPUART1 are the kernel clock selected by LPUART1SEL and provided
to UCLK input, and the **pclk4** bus interface clock.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **SBSEN**: SBS peripheral clock enable
   Set and reset by software.
   0: SBS peripheral clock disabled (default after reset)
   1: SBS peripheral clock enabled

Bit 0 Reserved, must be kept at reset value.
### 7.8.54 RCC APB5 clock enable register (RCC_APB5ENR)

Address offset: 0x144
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:5 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 4 GFXTIMEN: GFXTIM peripheral clock enable</td>
</tr>
<tr>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0: GFXTIM peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1: GFXTIM peripheral clock provided</td>
</tr>
<tr>
<td>Bit 3 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 2 DCMIPPEN: DCMIPP peripheral clock enable</td>
</tr>
<tr>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0: DCMIPP peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1: DCMIPP peripheral clock provided</td>
</tr>
<tr>
<td>Bit 1 LTDCEN: LTDC peripheral clock enable</td>
</tr>
<tr>
<td>Provides the pixel clock (ltdc_clk) to the LTDC block.</td>
</tr>
<tr>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0: LTDC peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1: LTDC peripheral clock provided to the LTDC block</td>
</tr>
<tr>
<td>Bit 0 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address offset: 0x144</th>
<th>Reset value: 0x0000 0000</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>Bit 31:5 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 4 GFXTIMEN: GFXTIM peripheral clock enable</td>
</tr>
<tr>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0: GFXTIM peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1: GFXTIM peripheral clock provided</td>
</tr>
<tr>
<td>Bit 3 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 2 DCMIPPEN: DCMIPP peripheral clock enable</td>
</tr>
<tr>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0: DCMIPP peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1: DCMIPP peripheral clock provided</td>
</tr>
<tr>
<td>Bit 1 LTDCEN: LTDC peripheral clock enable</td>
</tr>
<tr>
<td>Provides the pixel clock (ltdc_clk) to the LTDC block.</td>
</tr>
<tr>
<td>Set and reset by software.</td>
</tr>
<tr>
<td>0: LTDC peripheral clock disabled (default after reset)</td>
</tr>
<tr>
<td>1: LTDC peripheral clock provided to the LTDC block</td>
</tr>
<tr>
<td>Bit 0 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
### 7.8.55  **RCC AHBS low-power clock enable register (RCC_AHB5LPENR)**

**Address offset:** 0x15C  
**Reset value:** 0xF018 513F

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Default</th>
<th>Remarks</th>
</tr>
</thead>
</table>
| 31    | AXISRAMLPEN: AXISRAM[4:1] low-power peripheral clock enable | rw      | 0         | 0: AXISRAM[4:1] interface peripheral clock disabled in low-power mode  
1: AXISRAM[4:1] interface peripheral clock enabled in low-power mode (default after reset) |
| 30    | ITCMLPEN: ITCM low-power peripheral clock enable  | rw      | 0         | 0: ITCM interface peripheral clock disabled in low-power mode  
1: ITCM interface peripheral clock enabled in low-power mode (default after reset) |
| 29    | DTCM2LPEN: DTCM2 low-power peripheral clock enable | rw      | 0         | 0: DTCM2 interface peripheral clock disabled in low-power mode  
1: DTCM2 interface peripheral clock enabled in low-power mode (default after reset) |
| 28    | DTCM1LPEN: DTCM1 low-power peripheral clock enable | rw      | 0         | 0: DTCM1 interface peripheral clock disabled in low-power mode  
1: DTCM1 interface peripheral clock enabled in low-power mode (default after reset) |
| 27:21 | Reserved, must be kept at reset value            |         |           |                                              |
| 20    | GPU2DLucken: GPU2D low-power peripheral clock enable | rw      | 0         | 0: GPU2D interface clock peripheral disabled in low-power mode  
1: GPU2D interface clock peripheral enabled in low-power mode (default after reset) |
| 19    | GFXMMULPEN: GFXMMU low-power peripheral clock enable | rw      | 0         | 0: GFXMMU interface peripheral clock disabled in low-power mode  
1: GFXMMU interface peripheral clock enabled in low-power mode (default after reset) |
| 18:15 | Reserved, must be kept at reset value            |         |           |                                              |
Bit 14 **XSPIMLPEN**: XSPIM low-power peripheral clock enable
Set and reset by software.
0: XSPIM interface peripheral clock disabled in low-power mode
1: XSPIM interface peripheral clock enabled in low-power mode (default after reset)

Bit 13 Reserved, must be kept at reset value.

Bit 12 **XSPI2LPEN**: XSPI2 and MCE2 low-power peripheral clock enable
Set and reset by software.
The hardware prevents writing this bit if XSPICKP = 1.
0: XSPI2 and MCE2 peripheral clock disabled in low-power mode
1: XSPI2 and MCE2 peripheral clock enabled in low-power mode (default after reset)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1LPEN**: SDMMC1 and SDMMC1 delay low-power peripheral clock enable
Set and reset by software.
0: SDMMC1 and SDMMC1 delay peripheral clock disabled in low-power mode
1: SDMMC1 and SDMMC1 delay peripheral clock enabled in low-power mode (default after reset)

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **XSPI1LPEN**: XSPI1 and MCE1 low-power peripheral clock enable
Set and reset by software.
The hardware prevents writing this bit if XSPICKP = 1.
0: XSPI1 and MCE1 peripheral clock disabled in low-power mode
1: XSPI1 and MCE1 peripheral clock enabled in low-power mode (default after reset)

Bit 4 **FMCLPEN**: FMC and MCE3 peripheral clocks enable in low-power mode
Set and reset by software.
The hardware prevents writing this bit if FMCCKP = 1.
0: FMC and MCE3 peripheral clocks disabled in low-power mode
1: FMC and MCE3 peripheral clocks enabled in low-power mode (default after reset):
The peripheral clocks of the FMC are the kernel clock selected by FMCSEL, and the hclk5 bus interface clock.

Bit 3 **JPEGLPEN**: JPEG clock enable in low-power mode
Set and reset by software.
0: JPEG peripheral clock disabled in low-power mode
1: JPEG peripheral clock enabled in low-power mode (default after reset)

Bit 2 **FLASHLPEN**: FLASH low-power peripheral clock enable
Set and reset by software.
0: FLASH peripheral clock disabled in low-power mode
1: FLASH peripheral clock enabled in low-power mode (default after reset)

Bit 1 **DMA2DLPEN**: DMA2D low-power peripheral clock enable
Set and reset by software.
0: DMA2D peripheral clock disabled in low-power mode
1: DMA2D peripheral clock enabled in low-power mode (default after reset)

Bit 0 **HPDMA1LPEN**: HPDMA1 low-power peripheral clock enable
Set and reset by software.
0: HPDMA1 peripheral clock disabled in low-power mode
1: HPDMA1 peripheral clock enabled in low-power mode (default after reset)
### 7.8.56 RCC_AHB1 low-power clock enable register (RCC_AHB1LPENR)

**Address offset:** 0x160

**Reset value:** 0x8E03 8030

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
<th>Access</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ADF1LPEN</td>
<td>ADF clock enable in low-power mode</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>OTGFSLPEN</td>
<td>OTGFS clock enable in low-power mode</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>26</td>
<td>USBPHYCLPEN</td>
<td>USBPHYC peripheral clock enable in low-power mode</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>25</td>
<td>OTGHSLPEN</td>
<td>OTGHS peripheral clock enable in low-power mode</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>24</td>
<td>UCPDCTRL</td>
<td>USBPHYC common block power-down control</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>ETH1RXLPEN</td>
<td>ETH1 reception peripheral clock enable in low-power mode</td>
<td>rw</td>
<td>0x0</td>
</tr>
<tr>
<td>16</td>
<td>ETH1XLPM</td>
<td>ETH1 peripheral clock enable in low-power mode</td>
<td>rw</td>
<td>0x0</td>
</tr>
</tbody>
</table>

- **Bit 31** **ADF1LPEN**: ADF clock enable in low-power mode
  - Set and reset by software.
  - 0: ADF peripheral clock disabled in low-power mode
  - 1: ADF peripheral clock enabled in low-power mode (default after reset)

- **Bits 30:28** Reserved, must be kept at reset value.

- **Bit 27** **OTGFSLPEN**: OTGFS clock enable in low-power mode
  - Set and reset by software.
  - 0: OTGFS peripheral clock disabled in low-power mode
  - 1: OTGFS peripheral clock enabled in low-power mode (default after reset)

- **Bit 26** **USBPHYCLPEN**: USBPHYC peripheral clock enable in low-power mode
  - Set and reset by software.
  - 0: USBPHYC peripheral clock disabled in low-power mode
  - 1: USBPHYC peripheral clock enabled in low-power mode (default after reset)

- **Bit 25** **OTGHSLPEN**: OTGHS peripheral clock enable in low-power mode
  - Set and reset by software.
  - 0: OTGHS peripheral clock disabled in low-power mode
  - 1: OTGHS peripheral clock enabled in low-power mode (default after reset)

- **Bit 24** **UCPDCTRL**: USBPHYC common block power-down control
  - Set and reset by software.
  - 0: In SUSPEND, PHY state machine, bias and USBPHYC PLL remain powered (default after reset).
  - 1: In SUSPEND, PHY state machine, bias and USBPHYC PLL are powered down.

- **Bits 23:18** Reserved, must be kept at reset value.

- **Bit 17** **ETH1RXLPEN**: ETH1 reception peripheral clock enable in low-power mode
  - Set and reset by software.
  - 0: ETH1 reception peripheral clock disabled in low-power mode
  - 1: ETH1 reception peripheral clock enabled in low-power mode (default after reset)
Bit 16  **ETH1TXLPEN**: ETH1 transmission peripheral clock enable in low-power mode
   Set and reset by software.
   0: ETH1 transmission peripheral clock disabled in low-power mode
   1: ETH1 transmission peripheral clock enabled in low-power mode (default after reset)

Bit 15  **ETH1MACLPEN**: ETH1 MAC peripheral clock enable in low-power mode
   Set and reset by software.
   0: ETH1 MAC peripheral clock disabled in low-power mode
   1: ETH1 MAC peripheral clock enabled in low-power mode (default after reset)

Bits 14:6  Reserved, must be kept at reset value.

Bit 5  **ADC12LPEN**: ADC1 and 2 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: ADC1 and 2 peripheral clocks disabled in low-power mode
   1: ADC1 and 2 peripheral clocks enabled in low-power mode (default after reset)
   The peripheral clocks of the ADC1 and 2 are the kernel clock selected by ADCSEL and provided to
   ADCx_CK input, and the **rcc_hclk1** bus interface clock.

Bit 4  **GPDMA1LPEN**: GPDMA1 clock enable in low-power mode
   Set and reset by software.
   0: GPDMA1 clock disabled in low-power mode
   1: GPDMA1 clock enabled in low-power mode (default after reset)

Bits 3:0  Reserved, must be kept at reset value.
### 7.8.57 RCC AHB2 low-power clock enable register (RCC_AHB2LPENR)

Address offset: 0x164  
Reset value: 0x6000 4202

![Register Layout](image)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>SRAM2LPEN</td>
<td>SRAM2 clock enable in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SRAM2 clock disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SRAM2 clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>29</td>
<td>SRAM1LPEN</td>
<td>SRAM1 clock enable in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SRAM1 clock disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SRAM1 clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>28:15</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>CORDICLPEN</td>
<td>CORDIC clock enable in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: CORDIC clock disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CORDIC clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>13:10</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>SDMMC2LPEN</td>
<td>SDMMC2 and SDMMC2 delay clock enable in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SDMMC2 and SDMMC2 delay clock disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SDMMC2 and SDMMC2 delay clock enabled in low-power mode</td>
</tr>
<tr>
<td>8:2</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>PSSILPEN</td>
<td>PSSI peripheral clock enable in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: PSSI peripheral clock disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: PSSI peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
### 7.8.58 RCC AHB3 low-power clock enable register (RCC_AHB3LPENR)

Address offset: 0x16C  
Reset value: 0x0000 0057

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Setting and Reset by Software</th>
<th>Default (after Reset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-7</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td><strong>PKALPEN</strong>: PKA peripheral clock enable in low-power mode</td>
<td>Set and reset by software.</td>
<td>1: PKA peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td><strong>SAESLPEN</strong>: SAES peripheral clock enable in low-power mode</td>
<td>Set and reset by software.</td>
<td>1: SAES peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>CRYPLPEN</strong>: CRYP peripheral clock enable in low-power mode</td>
<td>Set and reset by software.</td>
<td>1: CRYP peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>1</td>
<td><strong>HASHLPEN</strong>: HASH peripheral clock enable in low-power mode</td>
<td>Set and reset by software.</td>
<td>1: HASH peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
<tr>
<td>0</td>
<td><strong>RNGLPEN</strong>: RNG peripheral clock enable in low-power mode</td>
<td>Set and reset by software.</td>
<td>1: RNG peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>
### 7.8.59 RCC AHB4 low-power clock enable register (RCC_AHB4LPENR)

Address offset: 0x168  
Reset value: 0x1008 F0FF

<table>
<thead>
<tr>
<th>Bit 31:29</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BKPRAMLPCEN</td>
<td>Backup RAM clock enable in low-power mode</td>
<td>1: Backup RAM clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27:20</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 19</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRCLPCEN</td>
<td>CRC clock enable in low-power mode</td>
<td>1: CRC clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 18:16</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIOPLPCEN</td>
<td>GPIO peripheral clock enable in low-power mode</td>
<td>1: GPIO peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIOOLPCEN</td>
<td>GPIOO peripheral clock enable in low-power mode</td>
<td>1: GPIOO peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 13</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIONLPCEN</td>
<td>GPION peripheral clock enable in low-power mode</td>
<td>1: GPION peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIOMLPCEN</td>
<td>GPIOM peripheral clock enable in low-power mode</td>
<td>1: GPIOM peripheral clock enabled in low-power mode (default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 11:8</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 7  **GPIOHLPEN**: GPIOH peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOH peripheral clock disabled in low-power mode
      1: GPIOH peripheral clock enabled in low-power mode (default after reset)

Bit 6  **GPIOGLPEN**: GPIOG peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOG peripheral clock disabled in low-power mode
      1: GPIOG peripheral clock enabled in low-power mode (default after reset)

Bit 5  **GPIOFLPEN**: GPIOF peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOF peripheral clock disabled in low-power mode
      1: GPIOF peripheral clock enabled in low-power mode (default after reset)

Bit 4  **GPIOELPEN**: GPIOE peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOE peripheral clock disabled in low-power mode
      1: GPIOE peripheral clock enabled in low-power mode (default after reset)

Bit 3  **GPIODLPEN**: GPIOD peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOD peripheral clock disabled in low-power mode
      1: GPIOD peripheral clock enabled in low-power mode (default after reset)

Bit 2  **GPIOCLPEN**: GPIOC peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOC peripheral clock disabled in low-power mode
      1: GPIOC peripheral clock enabled in low-power mode (default after reset)

Bit 1  **GPIOBLPEN**: GPIOB peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOB peripheral clock disabled in low-power mode
      1: GPIOB peripheral clock enabled in low-power mode (default after reset)

Bit 0  **GPIOALPEN**: GPIOA peripheral clock enable in low-power mode
      Set and reset by software.
      0: GPIOA peripheral clock disabled in low-power mode
      1: GPIOA peripheral clock enabled in low-power mode (default after reset)
## 7.8.60 RCC APB1 low-power clock enable register 1 (RCC_APB1LPENR1)

Address offset: 0x170
Reset value: 0xC8FF CBFF

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UART8LPEN: UART8 peripheral clocks enable in low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: UART8 peripheral clocks disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>1: UART8 peripheral clocks enabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>(default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>UART7LPEN: UART7 peripheral clocks enable in low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: UART7 peripheral clocks disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>1: UART7 peripheral clocks enabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>(default after reset)</td>
</tr>
</tbody>
</table>

| Bits 29:28 Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>CECLPEN: HDMI-CEC peripheral clocks enable in low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: HDMI-CEC peripheral clocks disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>1: HDMI-CEC peripheral clocks enabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>(default after reset)</td>
</tr>
</tbody>
</table>

| Bits 26:24 Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 23</th>
<th>I2C3LPEN: I2C3 peripheral clocks enable in low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: I2C3 peripheral clocks disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>1: I2C3 peripheral clocks enabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>(default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 22</th>
<th>I2C2LPEN: I2C2 peripheral clocks enable in low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: I2C2 peripheral clocks disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>1: I2C2 peripheral clocks enabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>(default after reset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 21</th>
<th>I2C1_I3C1LPEN: I2C1/I3C1 peripheral clocks enable in low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set and reset by software.</td>
</tr>
<tr>
<td></td>
<td>0: I2C1/I3C1 peripheral clocks disabled in low-power mode</td>
</tr>
<tr>
<td></td>
<td>1: I2C1/I3C1 peripheral clocks enabled in low-power mode</td>
</tr>
</tbody>
</table>
Bit 20 **UART5LPEN**: UART5 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: UART5 peripheral clocks disabled in low-power mode
   1: UART5 peripheral clocks enabled in low-power mode (default after reset)

Bit 19 **UART4LPEN**: UART4 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: UART4 peripheral clocks disabled in low-power mode
   1: UART4 peripheral clocks enabled in low-power mode (default after reset)

Bit 18 **USART3LPEN**: USART3 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: USART3 peripheral clocks disabled in low-power mode
   1: USART3 peripheral clocks enabled in low-power mode (default after reset)

Bit 17 **USART2LPEN**: USART2 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: USART2 peripheral clocks disabled in low-power mode
   1: USART2 peripheral clocks enabled in low-power mode (default after reset).

Bit 16 **SPDIFRXLPEN**: SPDIFRX peripheral clocks enable in low-power mode
   Set and reset by software.
   0: SPDIFRX peripheral clocks disabled in low-power mode
   1: SPDIFRX peripheral clocks enabled in low-power mode (default after reset).

Bit 15 **SPI3LPEN**: SPI3 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: SPI3 peripheral clocks disabled in low-power mode
   1: SPI3 peripheral clocks enabled in low-power mode (default after reset).

Bit 14 **SPI2LPEN**: SPI2 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: SPI2 peripheral clocks disabled in low-power mode
   1: SPI2 peripheral clocks enabled in low-power mode (default after reset).

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **WWDGLPEN**: WWDG clock enable in low-power mode
   Set and reset by software.
   0: WWDG clock disable in low-power mode
   1: WWDG clock enabled in low-power mode (default after reset)

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: LPTIM1 peripheral clocks disabled in low-power mode
   1: LPTIM1 peripheral clocks enabled in low-power mode (default after reset).

Bit 8 **TIM14LPEN**: TIM14 peripheral clock enable in low-power mode
   Set and reset by software.
   0: TIM14 peripheral clock disabled in low-power mode
   1: TIM14 peripheral clock enabled in low-power mode (default after reset)

Bit 7 **TIM13LPEN**: TIM13 peripheral clock enable in low-power mode
   Set and reset by software.
   0: TIM13 peripheral clock disabled in low-power mode
   1: TIM13 peripheral clock enabled in low-power mode (default after reset)
Bit 6 **TIM12LPEN**: TIM12 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM12 peripheral clock disabled in low-power mode
1: TIM12 peripheral clock enabled in low-power mode (default after reset)

Bit 5 **TIM7LPEN**: TIM7 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM7 peripheral clock disabled in low-power mode
1: TIM7 peripheral clock enabled in low-power mode (default after reset)

Bit 4 **TIM6LPEN**: TIM6 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM6 peripheral clock disabled in low-power mode
1: TIM6 peripheral clock enabled in low-power mode (default after reset)

Bit 3 **TIM5LPEN**: TIM5 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM5 peripheral clock disabled in low-power mode
1: TIM5 peripheral clock enabled in low-power mode (default after reset)

Bit 2 **TIM4LPEN**: TIM4 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM4 peripheral clock disabled in low-power mode
1: TIM4 peripheral clock enabled in low-power mode (default after reset)

Bit 1 **TIM3LPEN**: TIM3 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM3 peripheral clock disabled in low-power mode
1: TIM3 peripheral clock enabled in low-power mode (default after reset)

Bit 0 **TIM2LPEN**: TIM2 peripheral clock enable in low-power mode
Set and reset by software.
0: TIM2 peripheral clock disabled in low-power mode
1: TIM2 peripheral clock enabled in low-power mode (default after reset)
7.8.61   RCC APB1 low-power clock enable register 2 (RCC_APB1LPENR2)

Address offset: 0x174
Reset value: 0x0800 0122

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

* rw

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

* rw

Bits 31:28   Reserved, must be kept at reset value.

Bit 27   UCPD1LPEN: UCPD peripheral clock enable in low-power mode
Set and reset by software.
0: UCPD peripheral clock disabled in low-power mode
1: UCPD peripheral clock enabled in low-power mode (default after reset)

Bits 26:9   Reserved, must be kept at reset value.

Bit 8   FDCANLPEN: FDCAN peripheral clock enable in low-power mode
Set and reset by software.
0: FDCAN peripheral clock disabled in low-power mode
1: FDCAN peripheral clock enabled in low-power mode (default after reset)

Bits 7:6   Reserved, must be kept at reset value.

Bit 5   MDIOSLPEN: MDIOS peripheral clock enable in low-power mode
Set and reset by software.
0: MDIOS peripheral clock disabled in low-power mode
1: MDIOS peripheral clock enabled in low-power mode (default after reset)

Bits 4:2   Reserved, must be kept at reset value.

Bit 1   CRSLPEN: clock recovery system peripheral clock enable in low-power mode
Set and reset by software.
0: CRS peripheral clock disabled in low-power mode
1: CRS peripheral clock enabled in low-power mode (default after reset)

Bit 0   Reserved, must be kept at reset value.
### 7.8.62  RCC APB2 low-power clock enable register (RCC_APB2LPENR)

Address offset: 0x178

Reset value: 0x00DF 3011

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:24</td>
<td>Reserved, must be kept at reset value.</td>
<td>Bit 31:24 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
| 23   | **SAI2LPEN**: SAI2 peripheral clocks enable in low-power mode | Set and reset by software.  
0: SAI2 peripheral clocks disabled in low-power mode  
1: SAI2 peripheral clocks enabled in low-power mode (default after reset)  
The peripheral clocks of the SAI2 are: the kernel clock selected by SAI2SEL and provided to SAI_CK_A and SAI_CK_B inputs, and the pclk2 bus interface clock. |
| 22   | **SAI1LPEN**: SAI1 peripheral clocks enable in low-power mode | Set and reset by software.  
0: SAI1 peripheral clocks disabled in low-power mode  
1: SAI1 peripheral clocks enabled in low-power mode (default after reset)  
The peripheral clocks of the SAI1 are: the kernel clock selected by SAI1SEL and provided to SAI_CK_A and SAI_CK_B inputs, and the pclk2 bus interface clock. |
| 21   | Reserved, must be kept at reset value. | Bit 21 Reserved, must be kept at reset value.                          |
| 20   | **SPI5LPEN**: SPI5 peripheral clocks enable in low-power mode | Set and reset by software.  
0: SPI5 peripheral clocks disabled in low-power mode  
1: SPI5 peripheral clocks enabled in low-power mode (default after reset)  
The peripheral clocks of the SPI5 are the kernel clock selected by SPI45SEL and provided to com_clk input, and the pclk2 bus interface clock. |
| 19   | **TIM9LPEN**: TIM9 peripheral clock enable in low-power mode | Set and reset by software.  
0: TIM9 peripheral clock disabled in low-power mode  
1: TIM9 peripheral clock enabled in low-power mode (default after reset) |
| 18   | **TIM17LPEN**: TIM17 peripheral clock enable in low-power mode | Set and reset by software.  
0: TIM17 peripheral clock disabled in low-power mode  
1: TIM17 peripheral clock enabled in low-power mode (default after reset) |
Bit 17  **TIM16LPEN**: TIM16 peripheral clock enable in low-power mode  
Set and reset by software.  
0: TIM16 peripheral clock disabled in low-power mode  
1: TIM16 peripheral clock enabled in low-power mode (default after reset)

Bit 16  **TIM15LPEN**: TIM15 peripheral clock enable in low-power mode  
Set and reset by software.  
0: TIM15 peripheral clock disabled in low-power mode  
1: TIM15 peripheral clock enabled in low-power mode (default after reset)

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

Bit 13  **SPI4LPEN**: SPI4 peripheral clock enable in low-power mode  
Set and reset by software.  
0: SPI4 peripheral clocks disabled in low-power mode  
1: SPI4 peripheral clocks enabled in low-power mode (default after reset)  
The peripheral clocks of the SPI4 are: the kernel clock selected by SPI4SEL and provided to **com_clk** input, and the **pclk2** bus interface clock.

Bit 12  **SPI1LPEN**: SPI2S1 peripheral clock enable in low-power mode  
Set and reset by software.  
0: SPI2S1 peripheral clocks disabled in low-power mode  
1: SPI2S1 peripheral clocks enabled in low-power mode (default after reset)  
The peripheral clocks of the SPI2S1 are: the kernel clock selected by I2S1SEL and provided to **spi_ker_ck** input, and the **pclk2** bus interface clock.

Bits 11:5  Reserved, must be kept at reset value.

Bit 4  **USART1LPEN**: USART1 peripheral clock enable in low-power mode  
Set and reset by software.  
0: USART1 peripheral clocks disabled in low-power mode  
1: USART1 peripheral clocks enabled in low-power mode (default after reset)  
The peripheral clocks of the USART1 are the kernel clock selected by USART1SEL and provided to UCLK inputs, and the **pclk2** bus interface clock.

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **TIM1LPEN**: TIM1 peripheral clock enable in low-power mode  
Set and reset by software.  
0: TIM1 peripheral clock disabled in low-power mode  
1: TIM1 peripheral clock enabled in low-power mode (default after reset)
### 7.8.63 RCC APB4 low-power clock enable register (RCC_APB4LPENR)

Address offset: 0x17C  
Reset value: 0x0401 9E2A

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>DTSLPEN: temperature sensor peripheral clock enable in low-power mode</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: DTS peripheral clock disabled in low-power mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: DTS peripheral clock enabled in low-power mode (default after reset)</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>RTCAPBLPEN: RTC APB clock enable in low-power mode</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: The register clock interface of the RTC (APB) is disabled in low-power mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: The register clock interface of the RTC (APB) is enabled in low-power mode (default after reset)</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>VREFLPEN: VREF peripheral clock enable in low-power mode</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: VREF peripheral clock disabled in low-power mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: VREF peripheral clock enabled in low-power mode (default after reset)</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>LPTIM5LPEN: LPTIM5 peripheral clocks enable in low-power mode</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: LPTIM5 peripheral clocks disabled in low-power mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: LPTIM5 peripheral clocks enabled in low-power mode (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the LPTIM5 are the kernel clock selected by LPTIM45SEL and provided to clk_lpt input, and the pclk4 bus interface clock.</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>LPTIM4LPEN: LPTIM4 peripheral clocks enable in low-power mode</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Set and reset by software.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: LPTIM4 peripheral clocks disabled in low-power mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: LPTIM4 peripheral clocks enabled in low-power mode (default after reset)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The peripheral clocks of the LPTIM4 are the kernel clock selected by LPTIM45SEL and provided to clk_lpt input, and the pclk4 bus interface clock.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 10 LPTIM3LPEN: LPTIM3 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: LPTIM3 peripheral clocks disabled in low-power mode
   1: LPTIM3 peripheral clocks enabled in low-power mode (default after reset)
   The peripheral clocks of the LPTIM3 are the kernel clock selected by LPTIM23SEL and provided to clk_lpt input, and the pclk4 bus interface clock.

Bit 9 LPTIM2LPEN: LPTIM2 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: LPTIM2 peripheral clocks disabled in low-power mode
   1: LPTIM2 peripheral clocks enabled in low-power mode (default after reset)
   The peripheral clocks of the LPTIM2 are the kernel clock selected by LPTIM23SEL and provided to clk_lpt input, and the pclk4 bus interface clock.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 SPI6LPEN: SPI/I2S6 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: SPI/I2S6 peripheral clocks disabled in low-power mode
   1: SPI/I2S6 peripheral clocks enabled in low-power mode (default after reset)
   The peripheral clocks of the SPI/I2S6 are the kernel clock selected by SPI6SEL and provided to com_ck input, and the rcc_pclk4 bus interface clock.

Bit 4 Reserved, must be kept at reset value.

Bit 3 LPUART1LPEN: LPUART1 peripheral clocks enable in low-power mode
   Set and reset by software.
   0: LPUART1 peripheral clocks disabled in low-power mode
   1: LPUART1 peripheral clocks enabled in low-power mode (default after reset)
   The peripheral clocks of the LPUART1 are the kernel clock selected by LPUART1SEL and provided to UCLK input, and the rcc_pclk4 bus interface clock.

Bit 2 Reserved, must be kept at reset value.

Bit 1 SBSLPEN: SBS peripheral clock enable in low-power mode
   Set and reset by software.
   0: SBS peripheral clock disabled in low-power mode
   1: SBS peripheral clock enabled in low-power mode (default after reset)

Bit 0 Reserved, must be kept at reset value.
### 7.8.64 RCC APB5 low-power clock enable register (RCC_APB5LPENR)

**Address offset:** 0x180  
**Reset value:** 0x0000 0016

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:5</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 4    | **GFXTIMLPEN**: GFXTIM peripheral clock enable in low-power mode  
Set and reset by software.  
0: GFXTIM peripheral clock disabled in low-power mode  
1: GFXTIM peripheral clock enabled in low-power mode (default after reset) | rw             |               |
| 3    | Reserved, must be kept at reset value.          |                |               |
| 2    | **DCMIPPLPEN**: DCMIPP peripheral clock enable in low-power mode  
Set and reset by software.  
0: DCMIPP peripheral clock disabled in low-power mode  
1: DCMIPP peripheral clock enabled in low-power mode (default after reset) | rw             |               |
| 1    | **LTDCLPEN**: LTDC peripheral clock enable in low-power mode  
Set and reset by software.  
0: LTDC peripheral clock disabled in low-power mode  
1: LTDC peripheral clock enabled in low-power mode (default after reset) | rw             |               |
| 0    | Reserved, must be kept at reset value.          |                |               |
0x028

530/3791

RCC_PLLCKSELR

Reset value

DIVM3[5:0]

1 0 0 0 0 0

RM0477 Rev 6

DIVM2[5:0]

1 0 0 0 0 0

-

-

Reset value

DIVM1[5:0]

1 0 0 0 0 0

-

-

-

0 0 0

-

-

-

PLLSRC[1:0]

-

-

PPRE1[2:0]

-

Res.

-

Res.

-

Res.

-

Res.

Res.

-

-

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0 0 0

Res.

0 0 0

-

Res.

1 0 0 0 0 0
-

PPRE2[2:0]

RTCPRE[5:0]
STOPKERWUCK
STOPWUCK

-

Res.

Res.

-

Res.

Res.

-

Res.

-

Res.

Reset value

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

-

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

-

Res.

PPRE4[2:0]

Res.

Res.

Res.

Res.

Res.

TIMPRE

Res.

Res.

MCO1PRE[3:0]

MCO1SEL[2:0]

Res.

Res.

Res.

0 0

Res.

0 0 0
Res.

Res.

Res.

Res.

Res.

Res.

Res.

0

Res.

PPRE5[2:0]

Res.

Res.

Res.

Res.

Res.

Res.

0 0 0 0 0 0 0 0 0 0 0 0 0 0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

MCO2PRE[3:0]

1 0 0 0 0 0 0

Res.

Reset value
Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0 0 0 0 0

Res.

0x024

Res.

BMPRE[3:
0]

Res.

0x014

Res.

Res.

RCC_BMCFGR
Res.

Res.

Res.

CSITRIM[5:0]

Res.

0 0 0 0

Res.

Res.

Res.

HSITRIM[6:0]

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

Res.

Res.

0 0 0 0 0 0

Res.

RCC_APBCFGR

Res.

0x020

Res.

0x01C
RCC_CDCFGR

Res.

Reset value

Res.

Reset value

Res.

0x018
RCC_CFGR
MCO2SEL[2:0]

Reset value

Res.

0x010
RCC_CSICFGR

Res.

0x00C
RCC_CRRCR

Res.

Reset value

Res.

0x008
RCC_HSICFGR

Res.

0x004
HSIRDY
HSIKERON
HSION

HSIDIV[1:0]

HSIDIVF

Res.

CSIKERON
CSIRDY
CSION

Res.

Res.

HSI48RDY
HSI48ON

Res.

Res.

HSECSSON
HSEEXT
HSEBYP
HSERDY
HSEON

Res.

Res.

Res.

PLL3RDY
PLL3ON
PLL2RDY
PLL2ON
PLL1RDY
PLL1ON

Res.

Res.

RCC_CR

Res.

0x000
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0

Register Name

Res.

Offset

Res.

7.9

Res.

Reset and clock control (RCC)
RM0477

RCC register map
Table 73. RCC register map and reset values

1 0 0 1 0 1

HSICAL[11:0]

HSI48CAL[9:0]

CSICAL[7:0]

-

-

-

SWS[2: SW[2:0
0]
]

0 0 0 0 0 0 0 0 0 0 0 0 0 0

Reserved

CPRE[3:0]

0 0 0 0

0 0 0

Reserved

0 0


### Table 73. RCC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02C</td>
<td>RCC_PLLCFGR</td>
<td>PLL configuration and control register</td>
</tr>
<tr>
<td>0x030</td>
<td>RCC_PLL1DIVR1</td>
<td>PLL1 divider register</td>
</tr>
<tr>
<td>0x034</td>
<td>RCC_PLL1FRACR</td>
<td>PLL1 fractional divider register</td>
</tr>
<tr>
<td>0x038</td>
<td>RCC_PLL2DIVR1</td>
<td>PLL2 divider register</td>
</tr>
<tr>
<td>0x03C</td>
<td>RCC_PLL2FRACR</td>
<td>PLL2 fractional divider register</td>
</tr>
<tr>
<td>0x040</td>
<td>RCC_PLL3DIVR1</td>
<td>PLL3 divider register</td>
</tr>
<tr>
<td>0x044</td>
<td>RCC_PLL3FRACR</td>
<td>PLL3 fractional divider register</td>
</tr>
<tr>
<td>0x048</td>
<td>RCC_CCIPR1</td>
<td>Clock configuration and interface register 1</td>
</tr>
<tr>
<td>0x04C</td>
<td>RCC_CCIPR2</td>
<td>Clock configuration and interface register 2</td>
</tr>
</tbody>
</table>

**Reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02C</td>
<td>RCC_PLLCFGR</td>
<td>PLL configuration and control register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x030</td>
<td>RCC_PLL1DIVR1</td>
<td>PLL1 divider register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x034</td>
<td>RCC_PLL1FRACR</td>
<td>PLL1 fractional divider register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x038</td>
<td>RCC_PLL2DIVR1</td>
<td>PLL2 divider register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x03C</td>
<td>RCC_PLL2FRACR</td>
<td>PLL2 fractional divider register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x040</td>
<td>RCC_PLL3DIVR1</td>
<td>PLL3 divider register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x044</td>
<td>RCC_PLL3FRACR</td>
<td>PLL3 fractional divider register</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x048</td>
<td>RCC_CCIPR1</td>
<td>Clock configuration and interface register 1</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x04C</td>
<td>RCC_CCIPR2</td>
<td>Clock configuration and interface register 2</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
### Table 73. RCC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x054</td>
<td>RCC_CCIPR3</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Table 73. RCC register map and reset values

| Offset | Register Name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x07C  | RCC_AHB5RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x080  | RCC_AHB1RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x084  | RCC_AHB2RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x088  | RCC_AHB4RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x08C  | RCC_APB5RSTR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x090  | RCC_APB1RSTR1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x094  | RCC_APB1RSTR2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
## Table 73. RCC register map and reset values

| Offset  | Register Name      | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------|--------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x098   | RCC_APB2RSTR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x09C   | RCC_APB4RSTR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0A0   | Reserved           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0A4   | RCC_AHB3RSTR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0A8   | - 0x0AC            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0B0   | RCC_CKGDISR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0B4   | - 0x0BC            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C0   | RCC_PLL1DIVR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C4   | RCC_PLL2DIVR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C8   | RCC_PLL3DIVR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

**Reset values:**
- **RCC_APB2RSTR:** 00000000000000000000000000000000
- **RCC_APB4RSTR:** 00000000000000000000000000000000
- **RCC_AHB3RSTR:** 00000000000000000000000000000000
- **RCC_CKGDISR:** 10000000000000000000000000000000
- **RCC_PLL1DIVR2:** 00100000000000000000000000000000
- **RCC_PLL2DIVR2:** 00100000000000000000000000000000
- **RCC_PLL3DIVR2:** 00100000000000000000000000000000
### Table 73. RCC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
<th>Offset</th>
<th>Register Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0CC</td>
<td>RCC_PLL1SSCGR</td>
<td>0x0D0</td>
<td>RCC_PLL2SSCGR</td>
<td>0x0D4</td>
<td>RCC_PLL3SSCGR</td>
<td>0x0D8</td>
<td>-</td>
<td>0x0FC</td>
<td>Reserved</td>
<td>0x100</td>
<td>RCC_CKPROTR</td>
<td>0x104</td>
<td>-</td>
</tr>
<tr>
<td>0x130</td>
<td>RCC_RSR</td>
<td>0x134</td>
<td>RCC_AHB5ENR</td>
<td>0x138</td>
<td>-</td>
<td>0x140</td>
<td>-</td>
<td>0x14C</td>
<td>Reserved</td>
<td>0x150</td>
<td>-</td>
<td>0x154</td>
<td>-</td>
</tr>
</tbody>
</table>

#### Registers

- **RCC_PLL1SSCGR**: INCSTEP[14:0], SPREADSEL, MODPER[12:0]
- **RCC_PLL2SSCGR**: INCSTEP[14:0], SPREADSEL, MODPER[12:0]
- **RCC_PLL3SSCGR**: INCSTEP[14:0], SPREADSEL, MODPER[12:0]
- **RCC_CKPROTR**: FMCSWP[2:0], XSPI2SWP[2:0], XSPI1SWP[2:0], FMCCKP, XSPICKP
- **RCC_RSR**: UPWRSTF, WWDRSTF, IWDRSTF, SFRSTF, PORRSTF, PINRSTF, BORRSTF, OBLRSTF, RMWFI, GPM2DEN, GFXMMUEN, XSPIMEN, XSPI2EN, XSPI1EN, FMCEN, JPEGEN, DMA2DEN, HDMAMEN
- **RCC_AHB5ENR**: GPU2DEN, GFXMMUEN, XSPIMEN, XSPI2EN, XSPI1EN, FMCEN, JPEGEN, DMA2DEN, HDMAMEN

| Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   | Offset  | Register Name   |
|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|---------|-----------------|
| 0x0CC   | RCC_PLL1SSCGR   | 0x0D0   | RCC_PLL2SSCGR   | 0x0D4   | RCC_PLL3SSCGR   | 0x0D8   | -                | 0x0FC   | Reserved        | 0x100  | RCC_CKPROTR     | 0x104  | -                |
| 0x130   | RCC_RSR         | 0x134   | RCC_AHB5ENR      | 0x138   | -                | 0x140   | -                | 0x14C   | Reserved        | 0x150  | -                | 0x154  | -                |

#### Reset Value

- RCC_PLL1SSCGR: 00000000000000000000000000000000
- RCC_PLL2SSCGR: 00000000000000000000000000000000
- RCC_PLL3SSCGR: 00000000000000000000000000000000
- RCC_CKPROTR: 00000000000000000000000000000000
- RCC_RSR: 00000000000000000000000000000000
- RCC_AHB5ENR: 00000000000000000000000000000000
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>Bits 31:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x138</td>
<td>RCC_AHB1ENR</td>
<td>ADF1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTGFSREN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OTGHSREN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETHRXEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETHTXEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ETHMACEN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x13C</td>
<td>RCC_AHB2ENR</td>
<td>SRAM2EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SRAM1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CORDIEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SDMMC2EN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x140</td>
<td>RCC_AHB4ENR</td>
<td>BKPRAMEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CRCEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPIPEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPIONEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPIONEN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x144</td>
<td>RCC_APB5ENR</td>
<td>UARTR7EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR7EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR7EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM14EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM13EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM12EN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x148</td>
<td>RCC_APB1ENR1</td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UART1EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UARTR6EN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x14C</td>
<td>RCC_APB1ENR2</td>
<td>LPTIM5EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM4EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM3EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM2EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM2EN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x150</td>
<td>RCC_APB2ENR</td>
<td>SAI2EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPI5EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM9EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM17EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM16EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TIM15EN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0x154</td>
<td>RCC_APB4ENR</td>
<td>DTSEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RTI2EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM15EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM14EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM13EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM12EN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LPTIM11EN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0</td>
</tr>
</tbody>
</table>

Table 73. RCC register map and reset values
### Table 73. RCC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register Name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x158</td>
<td>RCC_AHB3ENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x15C</td>
<td>RCC_AHB5LPENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x160</td>
<td>RCC_AHB1LPENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x164</td>
<td>RCC_AHB2LPENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x168</td>
<td>RCC_AHB4LPENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x16C</td>
<td>RCC_AHB3LPENR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x170</td>
<td>RCC_APB1LPENR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Reset and clock control (RCC)
Refer to Section 2.3 on page 149 for the register boundary addresses.
8 System configuration, boot and security (SBS)

8.1 SBS introduction

The STM32H7Rx/7Sx devices feature a set of configuration registers located in the SBS. On top of various device configurations, this SBS peripheral controls key boot and security features, including debug.

8.2 SBS main features

- Boot control
  - Upon system reset, configure the Cortex-M7 boot address and the hide protection level (HDPL) depending on the device life cycle (open or close).
- Debug control
  - Control the opening of the device debug interface, ensuring the sequencing of events that guarantee the device security
- System configuration
  - Analog switch configuration management
  - I2C Fast-mode Plus configuration
  - Selection of the Ethernet PHY interface
  - I/O compensation cell management
  - Selection of source input to the EXTI inputs used for external interrupt/event detection
  - Configuration for Cortex-M7 FPU interrupts
  - Enable/disable the divider by 32 on ADC12 clock
  - Optional one wait-state added to all AXI RAMs when ECC = 0
- Security status
  - BKPRAM and PKA SRAM erase
- RSSCMD[15:0] in SBS_RSSCMR for applications to pass on a command to the RSS (root secure services) executed at the next reset
8.3 SBS functional description

8.3.1 SBS block diagram

Figure 76. SBS block diagram

8.3.2 SBS internal signals

The tables below detail the SBS pins and internal signals.

Table 74. SBS input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOT</td>
<td>Input</td>
<td>Boot from flash memory or bootloader.</td>
</tr>
</tbody>
</table>

Table 75. SBS internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sbs_pclk</td>
<td>Input</td>
<td>APB bus clock</td>
</tr>
<tr>
<td>sbs_dbgrst</td>
<td>Input</td>
<td>Debug configuration reset from DBGMCU</td>
</tr>
<tr>
<td>flash_state</td>
<td>Non-volatile security status input from Flash memory</td>
<td></td>
</tr>
<tr>
<td>sbs_hdpl</td>
<td>Output</td>
<td>Hide protection level information</td>
</tr>
<tr>
<td>sbs_initvtor</td>
<td>Output</td>
<td>Vector address for Cortex-M7. See Section 8.3.5 for more details.</td>
</tr>
<tr>
<td>dbg_req</td>
<td>Input</td>
<td>External host request to launch the debug authentication protocol. See Section 8.3.6 for more details.</td>
</tr>
</tbody>
</table>

8.3.3 SBS reset and clocks

The SBS configuration port is clocked by the APB bus clock. There is a general reset and a debug configuration reset controlled in DBGMCU.
### 8.3.4 SBS hide protection management

The HDPL is a monotonic counter incremented during the boot stages. It is reset to its default value only following a power-on reset or a system reset. This default value (0 or 1) depends on the device lifecycle, as defined in Table 76: SBS boot control.

The device uses HDPL information to automatically isolate code and its associated secrets (like keys) during the boot process. Incrementing HDPL ensures that private code and data for one boot stage cannot be directly accessible from later boot stages.

**Note:** The use of HDPL in the embedded Flash memory is detailed in Section 5: Embedded flash memory (FLASH).

The HDPL can take a 0 to 3 value. When reaching the value 3, HDPL keeps this value until reset. The current HDPL value is readable in HDPL bitfield in SBS_HDPLSR register.

**Incrementing HDPL**

The application must write 0x6A to INCR_HDPL bitfield in SBS_HDPLCR register to increment the HDPL by one.

After such increment, and before doing any subsequent action, the application must check if the HDPL has effectively been incremented, by reading SBS_HDPLSR.

### 8.3.5 SBS boot control

The figure below and Table 76 shows how the SBS uses multiple information to define the Cortex-M7 boot address (INITVTOR) and the HDPL before the CPU reset is released.

**Figure 77. SBS boot control**

![SBS boot control diagram](MSv55532V2)
The BOOT value coming from the external pin is latched upon reset release. This pin is in input mode during the complete reset phase, and then switches automatically in analog mode after reset is released.

The NVSTATE value comes from the Flash memory interface, following an automatic option-byte load sequence (see Section 5: Embedded flash memory (FLASH) for details).

### 8.3.6 SBS debug management

Device debug is allowed or not depending on the lifecycle of the device:
- For open devices (NVSTATE = OPEN), the debug is always allowed.
- For close devices (NVSTATE = CLOSE), the debug is disabled at boot but can be reopened during the boot after an authenticated debug sequence.

When debug is forbidden, mailbox access port, CM7 access port, and CPU debug interface are locked. In this situation, the debugger cannot access the CPU and no effective debug can be done. Refer to Section 66: Debug infrastructure for more details.

#### Authenticated debug sequence

1. External host requests to launch the debug authentication protocol, via the DBGMCU access port mailbox. The rest of the device is kept under reset.
2. SBS selects the ST RSS boot address and requests the CPU to be released from reset.
3. CPU running RSS library executes the debug authentication library in system Flash memory. If the device is closed, the access port mailbox is closed until RSS acknowledges the authentication sequence start request.
4. As soon as a debug certificate is accepted by the device, RSS programs the opening of the Cortex-M7 access port by writing to AP_UNLOCK bitfield in SBS_DBGCR register. RSS also programs the opening of the CPU debug interface by writing to DBG_UNLOCK bitfield in SBS_DBGCR register.
5. Above re-openings are effective only when the HDPL in SBS_HDPLSR register has a value equal or superior to the value programmed in DBG_AUTH_HDPL in SBS_DBGCR register. If DBG_AUTH_HDPL bitfield is kept at 0x0, the debug never opens (forbidden value).
Note: The debug authentication library in system Flash memory is only available when HDPL = 0 in SBS_HDPLSR. Only this code can perform the above steps 3 and 4.

At any time, the application via DBGMCU can reset the debug configuration in SBS_DBGCR register. See dedicated sections for details (TBC).

**Debug locking**

To ensure the debug configuration is not modified by an unauthorized code, any application can lock it by writing to DBGCFG_LOCK bitfield in SBS_DBGLOCKR register. Once the debug configuration is locked, the application can unlock it using DBGMCU (TBC).

### 8.3.7 SBS voltage booster for I/O analog switches

By setting BOOSTEN in SBS_PMCR, the application reduces the total harmonic distortion (THD) of the I/O analog switches when the supply voltage is below 2.7 V, to guarantee the same performance as with the full voltage range.

To avoid current consumption due to booster activation when \( V_{DDA} < 2.7 \) V and \( V_{DD} > 2.7 \) V, \( V_{DD} \) can be selected as supply voltage for the analog switches, by setting BOOSTVDDSEL in SBS_PMCR.

In this case, BOOSTEN must be cleared to avoid unexpected power consumption.

#### Table 77. Analog switches recommended configuration

<table>
<thead>
<tr>
<th>( V_{DDA} )</th>
<th>( V_{DD} )</th>
<th>BOOSTEN</th>
<th>BOOSTVDDSEL</th>
<th>Analog switch supply voltage</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \leq 2.7 ) V</td>
<td>( \leq 2.7 ) V</td>
<td>1</td>
<td>0</td>
<td>Booster</td>
</tr>
<tr>
<td>( &gt; 2.7 ) V</td>
<td>( &gt; 2.7 ) V</td>
<td>0</td>
<td>1</td>
<td>( V_{DD} )</td>
</tr>
<tr>
<td>( &gt; 2.7 ) V</td>
<td>( \leq 2.7 ) V</td>
<td>0</td>
<td>0</td>
<td>( V_{DDA} )</td>
</tr>
</tbody>
</table>

### 8.3.8 SBS I/O compensation cell management

The I/O compensation cell generates an 8-bit value for the I/O buffer (4 bits for N-MOS and 4 bits for P-MOS), that depends on PVT operating conditions (process, voltage, temperature). These bits are used to control the output impedance in the I/O buffer, and the slew rate of the I/O commutation (the \( t_{fall} \) and \( t_{rise} \) time), in order to reduce the I/O noise on power supply.

The CSI is used by the compensation cells and must be enabled before enabling the compensation cells.

As shown in *Figure 78*, the compensation cell is split into two blocks: one block to provide an optimal code for the current PVT, and one block to drive the block controlled by the software.
The compensation cell value can be read when the READY flag is set in SBS_CCCSR. With CODESEL in SBS_CCCSR, the application can select the value to apply between two options: the code from the cell or the code from SBS_CCSWVALR.

Three compensation cells are embedded in STM32H7Rx/7Sx devices:

- one for the I/Os supplied by VDD or VDDA (see Section 8.3.7)
- one for the dedicated I/Os of the XSPI1
- one for the dedicated I/Os of the XSPI2

By default, the compensation cells are disabled, and a fixed code is applied to all the I/Os.

**Note:** The compensation cell can be used only when 2.7 V ≤ VDDIOx ≤ 3.6 V or 1.71 V ≤ VDDIOx ≤ 2 V (see the figure below).

### 8.3.9 SBS registers access control

Some SBS registers can be accessed only if HDPL = 0 in SBS_HDPLSR (hence only accessible by the ST RSS code).

Some bitfields in SBS are security critical, reset in non-secure state (coded as 0xB4) or in secure state (any other values). For bitfields reset in non-secure state, when the secure state is selected writing any value but 0xB4, the bitfield is no more writable until a reset occurs.
As a general rule, it is recommended to write 0xC3 to select a secure state and read back 0x6A to acknowledge the write is effective.

Protected fields are the following:
- INCR_HDPL
- DBGCFG_LOCK

### 8.3.10 SBS error management

The SBS does not manage errors.

### 8.4 SBS interrupts

The SBS does not support interrupts.

### 8.5 SBS registers

#### 8.5.1 SBS boot status register (SBS_BOOTSR)

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>INITVTOR[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |
| INITVTOR[15:0] |

Bits 31:0 INITVTOR[31:0]: initial vector for Cortex-M7
This register includes the physical boot address used by the Cortex-M7 after reset.
## 8.5.2 SBS hide protection control register (SBS_HDPLCR)

Address offset: 0x010  
Reset value: 0x0000 00B4

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **INCR_HDPL[7:0]**: increment HDPL  
Write 0x6A to increment device HDPL by one.  
After a write, the register value reverts to its default value (0xB4).

## 8.5.3 SBS hide protection status register (SBS_HDPLSR)

Address offset: 0x014  
Reset value: 0x0000 00XX (see Section 8.3.5)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **HDPL[7:0]**: hide protection level  
This bitfield returns the current HDPL of the device.  
0xB4: HDPL0, corresponding to ST-RSS (default when device is CLOSE)  
0x51: HDPL1  
0x8A: HDPL2  
0x6F and other codes: HDPL3, corresponding to non-boot application.  
*Note*: The device state (open/close) is defined in FLASH_NVSTATER register of the embedded Flash memory.
8.5.4 SBS debug control register (SBS_DBGCR)

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**DBG_AUTH_HDPL[7:0]**

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:16 **DBG_AUTH_HDPL[7:0]:** authenticated debug hide protection level

Writing to this bitfield defines at which HDPL the authenticated debug opens.

- 0x51: HDPL1
- 0x8A: HDPL2
- 0x6F: HDPL3

*Note: Writing any other values is ignored. Reading any other value means the authenticated debug always fails.*

Bits 15:8 **DBG_UNLOCK[7:0]:** debug unlock

Write 0xB4 to this bitfield to open the debug when HDPL in SBS_HDPLSR equals to DBG_AUTH_HDPL in this register.

Bits 7:0 **AP_UNLOCK[7:0]:** access port unlock

Write 0xB4 to this bitfield to open the device access port.

8.5.5 SBS debug lock register (SBS_DBGLOCKR)

Address offset: 0x024
Reset value: 0x0000 00B4

This register is reset when sbs_dbgcrst is asserted.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**DBGCFG_LOCK[7:0]**

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

Bits 7:0 **DBGCFG_LOCK[7:0]:** debug configuration lock

Reading this bitfield returns 0x6A if the bitfield value is different from 0xB4.
0xB4: Writes to SBS_DBGCR allowed (default)
Other: Writes to SBS_DBGCR ignored

*Note: 0xC3 is the recommended value to lock the debug configuration using this bitfield.*
8.5.6  SBS RSS command register (SBS_RSSCMDR)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**RSCMD[15:0]**

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **RSCMD[15:0]: RSS command**

The application can use this bitfield to pass on a command to the RSS, executed at the next reset.

8.5.7  SBS product mode and configuration register (SBS_PMCR)

Address offset: 0x100
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**ETH_PHYSEL[2:0]**

| rw | rw | rw |

Bits 31:29  Reserved, must be kept at reset value.

Bit 28  **AXISRAM_WS**: AXISRAM wait state

Set this bit to add one wait state to all AXISRAMs when ECC = 0. When ECC = 1 there is one wait state by default.

0: No wait state added when accessing any AXISRAM with ECC = 0
1: One wait state added when accessing any AXISRAM with ECC = 0. In this case, refer to the datasheet for maximum frequency.

Bits 27:24  Reserved, must be kept at reset value.

Bits 23:21  **ETH_PHYSEL[2:0]**: Ethernet PHY interface selection

000: GMII or MII
100: RMII
Other: reserved
8.5.8 SBS FPU interrupt mask register (SBS_FPUIMR)

Address offset: 0x104
Reset value: 0x0000 001F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FPU_IE[5:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 5:0 **FPU_IE[5:0]**: FPU interrupt enable
Set and cleared by software to enable the Cortex-M7 FPU interrupts

xxxxxx1: Invalid operation interrupt enabled (xxxxxx0 to disable)
xxxx1x: Divide-by-zero interrupt enabled (xxxxx0x to disable)
xxx1xx: Underflow interrupt enabled (xxx0xx to disable)
xx1xxx: Overflow interrupt enabled (xx0xxx to disable)
x1xxxx: Input denormal interrupt enabled (x0xxxx to disable)
1xxxxx: Inexact interrupt enabled (0xxxxx to disable), disabled by default
## 8.5.9 SBS memory erase status register (SBS_MESR)

Address offset: 0x108  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15:8</td>
</tr>
<tr>
<td>Bit 7:0</td>
</tr>
<tr>
<td>MEF</td>
</tr>
</tbody>
</table>

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

**Bit 0 MEF: memory erase flag**  
This bit is set by hardware when BKPRAM and PKA SRAM erase is ongoing after a power-on reset or one tamper event (see Section 50: Tamper and backup registers (TAMP) for details).  
This bit is set when the erase is done.  
0: Automatic erase of BKPRAM and PKA RAM memories ongoing  
1: Automatic erase of BKPRAM and PKA RAM memories done

## 8.5.10 SBS I/O compensation cell control and status register (SBS_CCCSR)

Address offset: 0x110  
Reset value: 0x0000 0000

For more details on this register, refer to Section 8.3.7 and Section 8.3.8.

<table>
<thead>
<tr>
<th>Bit 31:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15:8</td>
</tr>
<tr>
<td>XSPI2_IOHSLV</td>
</tr>
<tr>
<td>Bit 7:0</td>
</tr>
<tr>
<td>XSPI2_COMP_RDY</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.
Bit 18 **XSPI2_IOHSLV**: XSPIM_P2 I/O high speed at low voltage

When this bit is set, the speed of the XSPIM_P2 I/Os is optimized when the device voltage is low.

This bit is active only if XSPI2_HSLV user option bit is set in FLASH. This bit must be used only if the device supply voltage is below 2.7 V. Setting this bit when $V_{DD}$ is higher than 2.7 V may be destructive.

0: No XSPIM_P2 I/O speed optimization when device voltage is low
1: XSPIM_P2 I/O speed optimized when device voltage is low, if XSPI2_HSLV user option is set (no effect otherwise)

Bit 17 **XSPI1_IOHSLV**: XSPIM_P1 I/O high speed at low voltage

When this bit is set, the speed of the XSPIM_P1 I/Os is optimized when the device voltage is low.

This bit is active only if XSPI1_HSLV user option bit is set in FLASH. This bit must be used only if the device supply voltage is below 2.7 V. Setting this bit when $V_{DD}$ is higher than 2.7 V may be destructive.

0: No XSPIM_P1 I/O speed optimization when device voltage is low
1: XSPIM_P1 I/O speed optimized when device voltage is low, if XSPI1_HSLV user option is set (no effect otherwise)

Bit 16 **IOHSLV**: I/O high speed at low voltage

When this bit is set, the speed of the I/Os is optimized when the device voltage is low.

This bit is active only if VDDIO_HSLV user option bit is set in FLASH. It must be used only if the device supply voltage is below 2.7 V. Setting this bit when $V_{DD}$ is higher than 2.7 V may be destructive.

0: No I/O speed optimization when device voltage is low
1: I/O speed optimized when device voltage is low, if VDDIO_HSLV user option is set (no effect otherwise)

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **XSPI2_COMP_RDY**: XSPIM_P2 compensation cell ready

This bit provides the status of the XSPIM_P2 compensation cell.

0: XSPIM_P2 I/O compensation cell not ready
1: XSPIM_P2 I/O compensation cell ready. The code value provided by the cell can be used

*Note*: The CSI clock is required for the compensation cell to work properly. The compensation cell ready bit (XSPI2_COMP_RDY) is not set if the CSI clock is not enabled.

Bit 9 **XSPI1_COMP_RDY**: XSPIM_P1 compensation cell ready

This bit provides the status of the XSPIM_P1 compensation cell.

0: XSPIM_P1 I/O compensation cell not ready
1: XSPIM_P1 I/O compensation cell ready. The code value provided by the cell can be used

*Note*: The CSI clock is required for the compensation cell to work properly. The compensation cell ready bit (XSPI1_COMP_RDY) is not set if the CSI clock is not enabled.

Bit 8 **COMP_RDY**: Compensation cell ready

This bit provides the status of the compensation cell.

0: I/O compensation cell not ready
1: I/O compensation cell ready. The code value provided by the cell can be used.

*Note*: The CSI clock is required for the compensation cell to work properly. The compensation cell ready bit (COMP_RDY) is not set if the CSI clock is not enabled.

Bits 7:6 Reserved, must be kept at reset value.
Bit 5  **XSPI2_COMP_CODESEL**: XSPIM_P2 compensation cell code selection  
This bit selects the code to be applied for the XSPIM_P2 I/O compensation cell.  
0: Code from the cell (available in the SBS_CCVALR)  
1: Code from the SBS software value register (SBS_CCSWVALR)

Bit 4  **XSPI2_COMP_EN**: XSPIM_P2 compensation cell enable  
Set this bit to enable the XSPIM_P2 compensation cell.  
0: XSPIM_P2 compensation cell disabled  
1: XSPIM_P2 compensation cell enabled

Bit 3  **XSPI1_COMP_CODESEL**: XSPIM_P1 compensation cell code selection  
This bit selects the code to be applied for the XSPIM_P1 I/O compensation cell.  
0: Code from the cell (available in the SBS_CCVALR)  
1: Code from the SBS software value register (SBS_CCSWVALR)

Bit 2  **XSPI1_COMP_EN**: XSPIM_P1 compensation cell enable  
Set this bit to enable the XSPIM_P1 compensation cell.  
0: XSPIM_P1 compensation cell disabled  
1: XSPIM_P1 compensation cell enabled

Bit 1  **COMP_CODESEL**: Compensation cell code selection  
This bit selects the code to be applied for the I/O compensation cell.  
0: Code from the cell (available in the SBS_CCVALR)  
1: Code from the SBS software value register (SBS_CCSWVALR)

Bit 0  **COMP_EN**: Compensation cell enable  
Set this bit to enable the compensation cell.  
0: Compensation cell disabled  
1: Compensation cell enabled
8.5.11  SBS compensation cell for I/Os value register (SBS_CCVALR)

Address offset: 0x114
Reset value: 0x0000 0088
Reset: system reset

For more details on this register, refer to Section 8.3.8.

| Bits 31:24 | Reserved, must be kept at reset value. |
| Bits 23:20 | **XSPI2_PSRC[3:0]**: XSPIM_P2 PMOS transistors slew-rate compensation |
| Bits 19:16 | **XSPI2_NSRC[3:0]**: XSPIM_P2 NMOS transistors slew-rate compensation |
| Bits 15:12 | **XSPI1_PSRC[3:0]**: XSPIM_P1 PMOS transistors slew-rate compensation |
| Bits 11:8 | **XSPI1_NSRC[3:0]**: XSPIM_P1 NMOS transistors slew-rate compensation |
| Bits 7:4 | **PSRC[3:0]**: PMOS transistors slew-rate compensation |
| Bits 3:0 | **NSRC[3:0]**: NMOS transistors slew-rate compensation |

This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P2 to compensate the PMOS transistors slew rate in the functional range if XSPI2_COMP_CODESEL = 0 in SBS_CCCSR register.

This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P2 to compensate the NMOS transistors slew rate in the functional range if XSPI2_COMP_CODESEL = 0 in SBS_CCCSR register.

This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P1 to compensate the PMOS transistors slew rate in the functional range if XSPI1_COMP_CODESEL = 0 in SBS_CCCSR register.

This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P1 to compensate the NMOS transistors slew rate in the functional range if XSPI1_COMP_CODESEL = 0 in SBS_CCCSR register.

This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the PMOS transistors slew rate in the functional range if COMP_CODESEL = 0 in SBS_CCCSR register.

This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the NMOS transistors slew rate in the functional range if COMP_CODESEL = 0 in SBS_CCCSR register.
8.5.12 SBS compensation cell for I/Os software value register (SBS_CCSWVALR)

Address offset: 0x118
Reset value: 0x0000 0088

For more details on this register, refer to Section 8.3.8.

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:20  **XSPI2_SW_PSRC[3:0]**: XSPIM_P2 software PMOS transistors slew-rate compensation
This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P2 to compensate the PMOS transistors slew rate in the functional range if XSPI2_COMP_CODESEL = 1 in SBS_CCCSR register.

Bits 19:16  **XSPI2_SW_NSRC[3:0]**: XSPIM_P2 software NMOS transistors slew-rate compensation
This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P2 to compensate the NMOS transistors slew rate in the functional range if XSPI2_COMP_CODESEL = 1 in SBS_CCCSR register.

Bits 15:12  **XSPI1_SW_PSRC[3:0]**: XSPIM_P1 software PMOS transistors slew-rate compensation
This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P1 to compensate the PMOS transistors slew rate in the functional range if XSPI1_COMP_CODESEL = 1 in SBS_CCCSR register.

Bits 11:8  **XSPI1_SW_NSRC[3:0]**: XSPIM_P1 software NMOS transistors slew-rate compensation
This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted by XSPIM_P1 to compensate the NMOS transistors slew rate in the functional range if XSPI1_COMP_CODESEL = 1 in SBS_CCCSR register.

Bits 7:4  **SW_PSRC[3:0]**: Software PMOS transistors slew-rate compensation
This bitfield returns the PMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the PMOS transistors slew rate in the functional range if COMP_CODESEL = 1 in SBS_CCCSR register.

Bits 3:0  **SW_NSRC[3:0]**: Software NMOS transistors slew-rate compensation
This bitfield returns the NMOS transistors slew-rate compensation value computed by the cell. It is interpreted to compensate the NMOS transistors slew rate in the functional range if COMP_CODESEL = 1 in SBS_CCCSR register.
8.5.13 SBS break lockup register (SBS_BKLOCKR)

Address offset: 0x120
Reset value: 0x0000 0088

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ARAMECC_BL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ARAMECC_BL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 ARAMECC_BL: AXISRAM1 ECC error break lock
Set this bit to enable and lock the connection between AXISRAM1 ECC double error detection flag and break inputs of TIM1/15/16/17 peripherals.
Once set, this bit is cleared only by a system reset.
0: AXISRAM1 ECC double error detection flag disconnected from TIM1/15/16/17 break inputs
1: AXISRAM1 ECC double error detection flag connected to TIM1/15/16/17 break inputs

Bit 22 Reserved, must be kept at reset value.

Bit 21 ARAMECC_BL: AXISRAM3 ECC error break lock
Set this bit to enable and lock the connection between AXISRAM3 ECC double error detection flag and break inputs of TIM1/15/16/17 peripherals.
Once set this bit is cleared only by a system reset.
0: AXISRAM3 ECC double error detection flag disconnected from TIM1/15/16/17 break inputs
1: AXISRAM3 ECC double error detection flag connected to TIM1/15/16/17 break inputs

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 ITCMECC_BL: ITCM ECC error break lock
Set this bit to enable and lock the connection between ITCM ECC double error detection flag and break inputs of TIM1/15/16/17 peripherals.
Once set, this bit is cleared only by a system reset.
0: ITCM ECC double error detection flag disconnected from TIM1/15/16/17 break inputs
1: ITCM ECC double error detection flag connected to TIM1/15/16/17 break inputs

Bit 13 DTMECC_BL: DTCM ECC error break lock
Set this bit to enable and lock the connection between DTCM ECC double error detection flag and break inputs of TIM1/15/16/17 peripherals.
Once set, this bit is cleared only by a system reset.
0: DTCM ECC double error detection flag disconnected from TIM1/15/16/17 break inputs
1: DTCM ECC double error detection flag connected to TIM1/15/16/17 break inputs

Note: The DTCM0 and DTCM1 are Or’ed to give DTMECC
Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **BKRAMECC_BL**: Backup RAM ECC error break lock
Set this bit to enable and lock the connection between backup RAM ECC double error detection flag and break inputs of TIM1/15/16/17 peripherals.
Once set, this bit is cleared only by a system reset.
0: Backup RAM ECC double error detection flag disconnected from TIM1/15/16/17 break inputs.
1: Backup RAM ECC double error detection flag connected to TIM1/15/16/17 break inputs.

Bit 6 **CM7LCKUP_BL**: Cortex-M7 lockup break lock
Set this bit to enable and lock the connection between the Cortex-M7 lockup (HardFault) output and break inputs of TIM1/15/16/17 peripherals.
Once set, this bit is cleared only by a system reset.
0: Cortex-M7 lockup output disconnected from TIM1/15/16/17 break inputs
1: Cortex-M7 lockup output connected to TIM1/15/16/17 break inputs

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **FLASH_ECC_BL**: Flash ECC error break lock
Set this bit to enable and lock the connection between embedded flash memory ECC double error detection flag and break inputs of TIM1/15/16/17 peripherals.
Once set, this bit is cleared only by a system reset.
0: FLASH ECC double error detection flag disconnected from TIM1/15/16/17 break inputs
1: FLASH ECC double error detection flag connected to TIM1/15/16/17 break inputs

Bit 2 **PVD_BL**: PVD break lock
This bit is set by SW and cleared only by a system reset. It can be used to enable and lock the connection to TIM1/8/15/16/17 break input as well as the PVDE and PLS[2:0] bitfields in the PWR_CR1 register.
Once set, this bit is cleared only by a system reset.
0: PVD interrupt disconnected from TIM1/8/15/16/17 break inputs. PVDE and PLS[2:0] bitfields can be programmed by the application.
1: PVD output connected to TIM1/8/15/16/17 break input. PVDE and PLS[2:0] bits are read only.

Bits 1:0 Reserved, must be kept at reset value.
## 8.5.14 SBS external interrupt configuration register x (SBS_EXTICRx)

Address offset: 0x130 + 0x4 * (x-1) (x = 1 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 15:0</td>
<td><strong>PC_EXTI</strong>(x-1) * 4 + i**: Port configuration EXTI (x * 4 + i) (i = 0 to 3)</td>
</tr>
<tr>
<td></td>
<td>This bitfield selects the source input to the EXTI input ((x-1) * 4 + i) used for external interrupt/ event detection.</td>
</tr>
<tr>
<td>0x0:</td>
<td>PA[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x1:</td>
<td>PB[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x2:</td>
<td>PC[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x3:</td>
<td>PD[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x4:</td>
<td>PE[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x5:</td>
<td>PF[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x6:</td>
<td>PG[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x7:</td>
<td>PH[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x8:</td>
<td>PM[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0x9:</td>
<td>PD[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0xA:</td>
<td>PO[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>0xB:</td>
<td>PP[(x-1)*4+i] pin</td>
</tr>
<tr>
<td>Others:</td>
<td>reserved</td>
</tr>
</tbody>
</table>
### Table 78. SBS register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | SBS_BOOTSR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  |
| 0x010  | SBS_HDPLCR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1  | 0  | 1  | 0  | 0  | 0  | 0  |
| 0x014  | SBS_HDPLSR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | X  | X  | X  | X  | X  | X  | X  |
| 0x018  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | SBS_DBGCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | SBS_DBGLOCKR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x034  | SBS_RSSCMDR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x038  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x100  | SBS_PMCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x104  | SBS_FPUIMR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x108  | SBS_MESR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x110  | SBS_CCCSR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x114  | SBS_CCVALR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x118  | SBS_CCWXVALR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.
Clock recovery system (CRS)

9.1 CRS introduction

The clock recovery system (CRS) is an advanced digital controller acting on the internal fine-granularity trimmable RC oscillator HSI48. The CRS provides powerful means to evaluate the oscillator output frequency, based on comparison with a selectable synchronization signal. The CRS can perform automatic trimming adjustments based on the measured frequency error value, while keeping the possibility of a manual trimming.

The CRS is ideally suited to provide a precise clock to the USB peripheral. In this case, the synchronization signal can be derived from the start-of-frame (SOF) packet signalization on the USB bus, sent by a USB host at 1 ms intervals.

The synchronization signal can also be derived from the LSE oscillator output, or generated by user software.

9.2 CRS main features

- Selectable synchronization source with programmable prescaler and polarity:
  - External pin
  - LSE oscillator output
  - USB SOF packet reception
- Possibility to generate synchronization pulses by software
- Automatic oscillator trimming capability with no need of CPU action
- Manual control option for faster startup convergence
- 16-bit frequency error counter with automatic error value capture and reload
- Programmable limit for automatic frequency error value evaluation and status reporting
- Maskable interrupts/events:
  - Expected synchronization (ESYNC)
  - Synchronization OK (SYNCOK)
  - Synchronization warning (SYNCWARN)
  - Synchronization or trimming error (ERR)

9.3 CRS implementation

<table>
<thead>
<tr>
<th>Feature</th>
<th>CRS</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIM width</td>
<td>6 bits</td>
</tr>
</tbody>
</table>
9.4 CRS functional description

9.4.1 CRS block diagram

Table 80. CRS internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crs_sync_in_1</td>
<td>Input</td>
<td>00: GPIO AF selected as SYNC signal source</td>
</tr>
<tr>
<td>crs_sync_in_2</td>
<td>Input</td>
<td>01:</td>
</tr>
<tr>
<td>crs_sync_in_3</td>
<td>Input</td>
<td>10: selected as SYNC signal source (default)</td>
</tr>
<tr>
<td>crs_sync_in_4</td>
<td>Input</td>
<td>11:</td>
</tr>
</tbody>
</table>

Figure 80. CRS block diagram
9.5 CRS internal signals

Table 81 gives the list of CRS internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crs_it</td>
<td>Digital output</td>
<td>CRS interrupt</td>
</tr>
<tr>
<td>crs_pclk</td>
<td>Digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>hsi48_ck</td>
<td>Digital input</td>
<td>HSI48 oscillator clock</td>
</tr>
<tr>
<td>crs_trim[0:5]</td>
<td>Digital output</td>
<td>HSI48 oscillator smooth trimming value</td>
</tr>
<tr>
<td>crs_sync0</td>
<td>Digital input</td>
<td>SYNC signal source selection (SYNC, or LSE, or OTG_HS, or OTG_FS)</td>
</tr>
<tr>
<td>crs_sync1</td>
<td>Digital input</td>
<td></td>
</tr>
<tr>
<td>crs_sync2</td>
<td>Digital input</td>
<td></td>
</tr>
<tr>
<td>crs_sync3</td>
<td>Digital input</td>
<td></td>
</tr>
</tbody>
</table>

9.5.1 Synchronization input

The CRS synchronization (SYNC) source, selectable through the CRS_CFGR register, can be the signal from an external signal (SYNC), the LSE clock, or the OTG HS SOF signal. This source signal also has a configurable polarity and can then be divided by a programmable binary prescaler to obtain a synchronization signal in a suitable frequency range (usually around 1 kHz).

For more information on the CRS synchronization source configuration, refer to Section 9.8.2.

It is also possible to generate a synchronization event by software, by setting the SWSYNC bit in the CRS_CR register.

9.5.2 Frequency error measurement

The frequency error counter is a 16-bit down/up counter, reloaded with the RELOAD value on each SYNC event. It starts counting down until it reaches the zero value, where the ESYNC (expected synchronization) event is generated. Then it starts counting up to the OUTRANGE limit, where it eventually stops (if no SYNC event is received), and generates a SYNCMISS event. The OUTRANGE limit is defined as the frequency error limit (FELIM field of the CRS_CFGR register) multiplied by 128.

When the SYNC event is detected, the actual value of the frequency error counter and its counting direction are stored in the FECAP (frequency error capture) field and in the FEDIR (frequency error direction) bit of the CRS_ISR register. When the SYNC event is detected during the down-counting phase (before reaching the zero value), it means that the actual frequency is lower than the target (the TRIM value must be incremented). When it is detected during the up-counting phase, it means that the actual frequency is higher (the TRIM value must be decremented).
9.5.3 Frequency error evaluation and automatic trimming

The measured frequency error is evaluated by comparing its value with a set of limits:
- **TOLERANCE LIMIT**, given directly in the FELIM field of the CRS_CFGR register
- **WARNING LIMIT**, defined as $3 \times FELIM$ value
- **OUTRANGE** (error limit), defined as $128 \times FELIM$ value

The result of this comparison is used to generate the status indication and also to control the automatic trimming, which is enabled by setting the AUTOTRIMEN bit in the CRS_CR register:
- When the frequency error is below the tolerance limit, it means that the actual trimming value in the TRIM field is the optimal one, hence no trimming action is needed.
  - SYNCOK status indicated
  - TRIM value not changed in AUTOTRIM mode
- When the frequency error is below the warning limit but above or equal to the tolerance limit, it means that some trimming action is necessary but that adjustment by one trimming step is enough to reach the optimal TRIM value.
  - SYNCOK status indicated
  - TRIM value adjusted by one trimming step in AUTOTRIM mode
- When the frequency error is above or equal to the warning limit but below the error limit, a stronger trimming action is necessary, and there is a risk that the optimal TRIM value is not reached for the next period.
  - SYNCWARN status indicated
  - TRIM value adjusted by two trimming steps in AUTOTRIM mode
- When the frequency error is above or equal to the error limit, the frequency is out of the trimming range. This can also happen when the SYNC input is not clean, or when some SYNC pulse is missing (for example when one USB SOF is corrupted).
  - SYNCERR or SYNCFMISS status indicated
  - TRIM value not changed in AUTOTRIM mode

**Note:**
If the actual value of the TRIM field is close to its limits and the automatic trimming can force it to overflow or underflow, the TRIM value is set to the limit, and the TRIMOVF status is indicated.

In AUTOTRIM mode (AUTOTRIMEN bit set in the CRS_CR register), the TRIM field of CRS_CR is adjusted by hardware and is read-only.

### 9.5.4 CRS initialization and configuration

**RELOAD value**

The RELOAD value must be selected according to the ratio between the target frequency and the frequency of the synchronization source after prescaling. This value is decreased by 1, to reach the expected synchronization on the zero value. The formula is the following:

\[
\text{RELOAD} = \left( \frac{f_{TARGET}}{f_{SYNC}} \right) - 1
\]

The reset value of the RELOAD field corresponds to a target frequency of 48 MHz and a synchronization signal frequency of 1 kHz (SOF signal from USB).

**FELIM value**

The selection of the FELIM value is closely coupled with the HSI48 oscillator characteristics and its typical trimming step size. The optimal value corresponds to half of the trimming step size, expressed as a number of oscillator clock ticks. The following formula can be used:

\[
\text{FELIM} = \left( \frac{f_{TARGET}}{f_{SYNC}} \right) \times \text{STEP} \% / 100 \% / 2
\]

The result must be always rounded up to the nearest integer value to obtain the best trimming response. If frequent trimming actions are not needed in the application, the hysteresis can be increased by slightly increasing the FELIM value.

The reset value of the FELIM field corresponds to \( \left( \frac{f_{TARGET}}{f_{SYNC}} \right) = 48000 \), and to a typical trimming step size of 0.14%.

**Note:**
The trimming step size depends upon the product, check the datasheet for accurate setting.

**Caution:**
There is no hardware protection from a wrong configuration of the RELOAD and FELIM fields, this can lead to an erratic trimming response. The expected operational mode requires proper setup of the RELOAD value (according to the synchronization source frequency), which is also greater than 128 * FELIM value (OUTRANGE limit).
9.6 CRS in low-power modes

Table 82. Effect of low-power modes on CRS

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. CRS interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>CRS registers are frozen. The CRS stops operating until the Stop mode is exited and the HSI48 oscillator is restarted.</td>
</tr>
<tr>
<td>Standby</td>
<td>The peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
<tr>
<td>Shutdown</td>
<td>The peripheral is powered down and must be reinitialized after exiting Shutdown mode.</td>
</tr>
</tbody>
</table>

9.7 CRS interrupts

Table 83. Interrupt control bits

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear flag bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expected synchronization</td>
<td>ESYNCF</td>
<td>ESYNCE</td>
<td>ESYNCC</td>
</tr>
<tr>
<td>Synchronization OK</td>
<td>SYNCKF</td>
<td>SYNCKIE</td>
<td>SYNCKOK</td>
</tr>
<tr>
<td>Synchronization warning</td>
<td>SYNCWARNF</td>
<td>SYNCWARNIE</td>
<td>SYNCWARNOK</td>
</tr>
<tr>
<td>(TRIMOVF, SYNCMISS, SYNCERR)</td>
<td>ERRF</td>
<td>ERRIE</td>
<td>ERRC</td>
</tr>
</tbody>
</table>

9.8 CRS registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed only by words (32-bit).

9.8.1 CRS control register (CRS_CR)

Address offset: 0x00
Reset value: 0x0000 2000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bits 13:8 **TRIM[5:0]**: HSI48 oscillator smooth trimming

These bits provide a user-programmable trimming value to the HSI48 oscillator. They can be programmed to adjust to variations in voltage and temperature that influence the oscillator frequency.

The default value is 32, corresponding to the middle of the trimming interval. The trimming step is specified in the product datasheet. A higher TRIM value corresponds to a higher output frequency.

When the AUTOTRIMEN bit is set, this field is controlled by hardware and is read-only.

Bit 7 **SWSYNC**: Generate software SYNC event

This bit is set by software in order to generate a software SYNC event. It is automatically cleared by hardware.

0: No action
1: A software SYNC event is generated.

Bit 6 **AUTOTRIMEN**: Automatic trimming enable

This bit enables the automatic hardware adjustment of TRIM bits according to the measured frequency error between two SYNC events. If this bit is set, the TRIM bits are read-only. The TRIM value can be adjusted by hardware by one or two steps at a time, depending on the measured frequency error value. Refer to Section 9.5.3 for more details.

0: Automatic trimming disabled, TRIM bits can be adjusted by the user.
1: Automatic trimming enabled, TRIM bits are read-only and under hardware control.

Bit 5 **CEN**: Frequency error counter enable

This bit enables the oscillator clock for the frequency error counter.

0: Frequency error counter disabled
1: Frequency error counter enabled

When this bit is set, the CRS_CFGR register is write-protected and cannot be modified.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **ESYNCE**: Expected SYNC interrupt enable

0: Expected SYNC (ESYNCF) interrupt disabled
1: Expected SYNC (ESYNCF) interrupt enabled

Bit 2 **ERRIE**: Synchronization or trimming error interrupt enable

0: Synchronization or trimming error (ERRF) interrupt disabled
1: Synchronization or trimming error (ERRF) interrupt enabled

Bit 1 **SYNCEWANIE**: SYNC warning interrupt enable

0: SYNC warning (SYNCWARNF) interrupt disabled
1: SYNC warning (SYNCWARNF) interrupt enabled

Bit 0 **SYNCKIE**: SYNC event OK interrupt enable

0: SYNC event OK (SYNCOKF) interrupt disabled
1: SYNC event OK (SYNCOKF) interrupt enabled
9.8.2 CRS configuration register (CRS_CFGR)

This register can be written only when the frequency error counter is disabled (the CEN bit is cleared in CRS_CR). When the counter is enabled, this register is write-protected.

Address offset: 0x04

Reset value: 0x2022 BB7F

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>SYNCPOL: SYNC polarity selection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software to select the input polarity for the SYNC signal source.</td>
</tr>
<tr>
<td>0</td>
<td>SYNC active on rising edge (default)</td>
</tr>
<tr>
<td>1</td>
<td>SYNC active on falling edge</td>
</tr>
</tbody>
</table>

| Bit 30 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 29:28</th>
<th>SYNCSRC[1:0]: SYNC signal source selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>CRS_SYNC pin selected as SYNC signal source</td>
</tr>
<tr>
<td>01</td>
<td>LSE selected as SYNC signal source</td>
</tr>
<tr>
<td>10</td>
<td>OTG FS SOF selected as SYNC signal source (default)</td>
</tr>
<tr>
<td>11</td>
<td>OTG HS SOF selected as SYNC signal source</td>
</tr>
</tbody>
</table>

**Note:** When using USB LPM (link power management) and the device is in Sleep mode, the periodic USB SOF is not generated by the host. No SYNC signal is therefore provided to the CRS to calibrate the HSI48 oscillator on the run. To guarantee the required clock precision after waking up from Sleep mode, the LSE clock or the SYNC pin must be used as SYNC signal.

| Bit 27 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 26:24</th>
<th>SYNCDIV[2:0]: SYNC divider</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>SYNC not divided (default)</td>
</tr>
<tr>
<td>001</td>
<td>SYNC divided by 2</td>
</tr>
<tr>
<td>010</td>
<td>SYNC divided by 4</td>
</tr>
<tr>
<td>011</td>
<td>SYNC divided by 8</td>
</tr>
<tr>
<td>100</td>
<td>SYNC divided by 16</td>
</tr>
<tr>
<td>101</td>
<td>SYNC divided by 32</td>
</tr>
<tr>
<td>110</td>
<td>SYNC divided by 64</td>
</tr>
<tr>
<td>111</td>
<td>SYNC divided by 128</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:16</th>
<th>FELIM[7:0]: Frequency error limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FELIM contains the value to be used to evaluate the captured frequency error value latched in the FECAP[15:0] bits of the CRS_ISR register. Refer to Section 9.5.3 for more details about FECAP evaluation.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>RELOAD[15:0]: Counter reload value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELOAD is the value to be loaded in the frequency error counter with each SYNC event. Refer to Section 9.5.2 for more details about counter behavior.</td>
<td></td>
</tr>
</tbody>
</table>
**9.8.3 CRS interrupt and status register (CRS_ISR)**

Address offset: 0x08  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>FECAP[15:0]</th>
<th>Frequency error capture</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>r</td>
<td>Frequency error capture</td>
</tr>
</tbody>
</table>

FECAP is the frequency error counter value latched in the time of the last SYNC event. Refer to Section 9.5.3 for more details about FECAP usage.

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>FEDIR</th>
<th>Frequency error direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>r</td>
<td>Up-counting direction, the actual frequency is above the target</td>
</tr>
</tbody>
</table>

0: Up-counting direction, the actual frequency is above the target  
1: Down-counting direction, the actual frequency is below the target

<table>
<thead>
<tr>
<th>Bit 14:11</th>
<th>TRIMOVF</th>
<th>Trimming overflow or underflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:11</td>
<td>r</td>
<td>Trimming overflow or underflow</td>
</tr>
</tbody>
</table>

This flag is set by hardware when the automatic trimming tries to over- or under-flow the TRIM value. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

<table>
<thead>
<tr>
<th>Bit 9</th>
<th>SYNCMISS</th>
<th>SYNMC missed</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>r</td>
<td>SYNMC missed</td>
</tr>
</tbody>
</table>

This flag is set by hardware when the frequency error counter reaches value FELIM * 128 and no SYNC is detected, meaning either that a SYNC pulse was missed, or the frequency error is too big (internal frequency too high) to be compensated by adjusting the TRIM value, hence some other action must be taken. At this point, the frequency error counter is stopped (waiting for a next SYNC), and an interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

<table>
<thead>
<tr>
<th>Bit 8</th>
<th>SYNCERR</th>
<th>SYNC error</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>r</td>
<td>SYNC error</td>
</tr>
</tbody>
</table>

This flag is set by hardware when the SYNC pulse arrives before the ESYNC event and the measured frequency error is greater than or equal to FELIM * 128. This means that the frequency error is too big (internal frequency too low) to be compensated by adjusting the TRIM value, and that some other action has to be taken. An interrupt is generated if the ERRIE bit is set in the CRS_CR register. It is cleared by software by setting the ERRC bit in the CRS_ICR register.

<table>
<thead>
<tr>
<th>Bit 7:4</th>
<th>Reserved</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>r</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Reserved, must be kept at reset value.
9.8.4 CRS interrupt flag clear register (CRS_ICR)

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ESYNCC</td>
<td>ERYC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

**Bit 3 ESYNCC**: Expected SYNC clear flag
Writing 1 to this bit clears the ESYNCF flag in the CRS_ISR register.

**Bit 2 ERRC**: Error clear flag
Writing 1 to this bit clears TRIMOVF, SYNCMISS, and SYNCERR bits and consequently also the ERRF flag in the CRS_ISR register.
9.8.5 CRS register map

Table 84. CRS register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRS_CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x04</td>
<td>CRS_CFGR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x08</td>
<td>CRS_ISR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0C</td>
<td>CRS_ICR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
10 General-purpose I/Os (GPIO)

10.1 Introduction
Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16 bits reset register (GPIOx_BRR) and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

10.2 GPIO main features
- Output states: push-pull or open drain + pull-up/down
- Output data from output data register (GPIOx_ODR) or peripheral (alternate function output)
- Speed selection for each I/O
- Input states: floating, pull-up/down, analog
- Input data to input data register (GPIOx_IDR) or peripheral (alternate function input)
- Bit set and reset register (GPIOx_BSRR) for bitwise write access to GPIOx_ODR
- Lock mechanism (GPIOx_LCKR) provided to freeze the I/O port configurations
- Analog function
- Alternate function selection registers
- Fast toggle capable of changing every two clock cycles
- Highly flexible pin multiplexing allows the use of I/O pins as GPIOs or as one of several peripheral functions

10.3 GPIO functional description
Subject to the specific hardware characteristics of each I/O port listed in the datasheet, each port bit of the general-purpose I/O (GPIO) ports can be individually configured by software in several modes:
- Input floating
- Input pull-up
- Input-pull-down
- Analog
- Output open-drain with pull-up or pull-down capability
- Output push-pull with pull-up or pull-down capability
- Alternate function push-pull with pull-up or pull-down capability
- Alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words or bytes. The GPIOx_BSRR and GPIOx_BRR registers allow atomic read/modify accesses to any of the GPIOx_ODR registers. In this way, there is no risk of an IRQ occurring between the read and the modify access.
The figure below shows the basic structure of a three-volt or five-volt tolerant GPIO (TT or FT). The Table 85 gives the possible port bit configurations.

**Figure 82. Structure of three-volt or five-volt tolerant GPIO (TT or FT)**

![Figure 82. Structure of three-volt or five-volt tolerant GPIO (TT or FT)](MSv46873V1)

Note: On a TT GPIO, the analog switch is not present and replaced by a direct connection. The analog bloc parasitic circuitry does not allow five-volt tolerance.

**Table 85. Port bit configuration**

<table>
<thead>
<tr>
<th>MODE(i) [1:0]</th>
<th>OTYPE(i) [1:0]</th>
<th>OSPEED(i) [1:0]</th>
<th>PUPD(i) [1:0]</th>
<th>I/O configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GP output PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>GP output PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>GP output PP + PD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>GP output OD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>GP output OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>GP output OD + PD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>Reserved (GP output OD)</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 85. Port bit configuration (continued)

<table>
<thead>
<tr>
<th>MODE[i] [1:0]</th>
<th>OTYPE[i]</th>
<th>OSPEED[i] [1:0]</th>
<th>PUPD[i] [1:0]</th>
<th>I/O configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0</td>
<td>0 0</td>
<td>AF</td>
<td>PP</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0 1</td>
<td>AF</td>
<td>PP + PU</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1 0</td>
<td>AF</td>
<td>PP + PD</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1 1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0 0</td>
<td>AF</td>
<td>OD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0 1</td>
<td>AF</td>
<td>OD + PU</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 0</td>
<td>AF</td>
<td>OD + PD</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1 1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>x</td>
<td>x x</td>
<td>0 0 Input</td>
<td>Floating</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>0 1 Input</td>
<td>PU</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 0 Input</td>
<td>PD</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 1 Reserved</td>
<td>(input floating)</td>
</tr>
<tr>
<td>11</td>
<td>x</td>
<td>x x</td>
<td>0 0 Input/output</td>
<td>Analog</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>0 1 Reserve</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 0 Reserved</td>
<td></td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>x x</td>
<td>1 1 Reserved</td>
<td></td>
</tr>
</tbody>
</table>

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

#### 10.3.1 General-purpose I/O (GPIO)

During and just after reset, the alternate functions are not active and most of the I/O ports are configured in analog mode.

The debug pins are in AF pull-up/pull-down after reset:
- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB3: JTDO/TRACESWO in floating state no pull-up/pull-down

When the pin is configured as output, the value written to the output data register (GPIOx_ODR) is output on the I/O pin. It is possible to use the output driver in push-pull mode or open-drain mode (only the low level is driven, high level is HI-Z).

The input data register (GPIOx_IDR) captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, that can be activated or not depending on the value in the GPIOx_PUPDR register.
10.3.2 I/O pin alternate function multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer that allows only one peripheral alternate function (AF) connected to an I/O pin at a time. In this way, there is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 alternate function inputs (AF0 to AF15) that can be configured through the GPIOx_AFRL (for pin 0 to 7) and GPIOx_AFRH (for pin 8 to 15) registers:

- After reset, the multiplexer selection is alternate function 0 (AF0). The I/Os are configured in alternate function mode through GPIOx_MODER register.
- The specific alternate function assignments for each pin are detailed in the device datasheet.

In addition to this flexible I/O multiplexing architecture, each peripheral has alternate functions mapped onto different I/O pins to optimize the number of peripherals available in smaller packages. The I/O pins on Port P and port O do not have alternate functions as these are dedicated for xSPI interface.

To use an I/O in a given configuration, the user must proceed as follows:

- **Debug function**: after each device reset these pins are assigned as alternate function pins immediately usable by the debugger host.
- **GPIO**: configure the desired I/O as output, input or analog in the GPIOx_MODER register.
- **Peripheral alternate function**:
  - Connect the I/O to the desired AFx in one of the GPIOx_AFRL or GPIOx_AFRH register.
  - Select the type, pull-up/pull-down and output speed via the GPIOx_OTYPER, GPIOx_PUPDR and GPIOx_OSPEEDR registers respectively.
  - Configure the desired I/O as an alternate function in the GPIOx_MODER register.
- **Additional functions**:
  - For all ADCs, configure the desired I/O in analog mode in the GPIOx_MODER register and configure the required function in the ADC registers.
  - For the additional functions like RTC, WKUPx and oscillators, configure the required function in the related RTC, PWR and RCC registers. These functions have priority over the configuration in the standard GPIO registers.

Refer to the “Alternate function mapping” table in the device datasheet for the detailed mapping of the alternate function I/O pins.

10.3.3 I/O port control registers

Each of the GPIO ports has four 32-bit memory-mapped control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR) to configure up to 16 I/Os. The GPIOx_MODER register is used to select the I/O mode (input, output, AF, analog). The GPIOx_OTYPER and GPIOx_OSPEEDR registers are used to select the output type (push-pull or open-drain) and speed. The GPIOx_PUPDR register is used to select the pull-up/pull-down whatever the I/O direction.
10.3.4 I/O port data registers

Each GPIO has two 16-bit memory-mapped data registers: input and output data registers (GPIO port input data register (GPIOx_IDR) (x = A to G, M to P) and GPIO port output data register (GPIOx_ODR) (x = A to G, M to P)).

GPIOx_ODR stores the data to be output, it is read/write accessible. The data input through the I/O are stored into the input data register (GPIOx_IDR), a read-only register.

10.3.5 I/O data bitwise handling

The bit set reset register (GPIOx_BSRR) is a 32-bit register that allows the application to set and reset each individual bit in the output data register (GPIOx_ODR). The bit set reset register has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i) bit. When written to 1, BR(i) resets the ODR(i) corresponding bit.

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using the GPIOx_BSRR register to change the values of individual bits in GPIOx_ODR is a “one-shot” effect that does not lock the GPIOx_ODR bits. The GPIOx_ODR bits can always be accessed directly. The GPIOx_BSRR register provides a way of performing atomic bitwise handling.

There is no need for the software to disable interrupts when programming the GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

10.3.6 GPIO locking mechanism

The GPIO control registers can be frozen by applying a specific write sequence to the GPIOx_LCKR register. The frozen registers are GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL, GPIOx_AFRH.

To write the GPIOx_LCKR register, a specific write/read sequence must be applied. When the right LOCK sequence is applied to the bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same). When the LOCK sequence is applied to a port bit, the value of the port bit can no longer be modified until the next MCU reset or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_AFRL and GPIOx_AFRH.

The LOCK sequence can only be performed using a word (32-bit long) access to the GPIOx_LCKR register due to the fact that GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

10.3.7 I/O alternate function input/output

Two registers are provided to select one of the alternate function inputs/outputs available for each I/O. With these registers, the user can connect an alternate function to some other pin as required by the application.

This means that a number of possible peripheral functions are multiplexed on each GPIO using the GPIOx_AFRL and GPIOx_AFRH alternate function registers. The application can
thus select any one of the possible functions for each I/O. The AF selection signal being common to the alternate function input and alternate function output, a single channel is selected for the alternate function input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the device datasheet.

### 10.3.8 External interrupt/wakeup lines

All ports have external interrupt capability. To use external interrupt lines, the port can be configured in input, output or alternate function mode (the port must not be configured in analog mode). Refer to Section 20: Extended interrupt and event controller (EXTI).

### 10.3.9 Input configuration

When the I/O port is programmed as input:
- The output buffer is disabled.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register provides the I/O state.

The figure below shows the input configuration of the I/O port bit.

![Input floating/pull-up/pull-down configurations](https://example.com/input-config-diagram.png)

**Figure 83. Input floating/pull-up/pull-down configurations**
10.3.10 Output configuration

When the I/O port is programmed as output:

- The output buffer is enabled:
  - Open-drain mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register leaves the port in Hi-Z (the P-MOS is never activated).
  - Push-pull mode: a 0 in the output register activates the N-MOS whereas a 1 in the output register activates the P-MOS.
- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

The figure below shows the output configuration of the I/O port bit.

Figure 84. Output configuration

10.3.11 Alternate function configuration

When the I/O port is programmed as alternate function:

- The output buffer can be configured in open-drain or push-pull mode.
- The output buffer is driven by the signals coming from the peripheral (transmitter enable and data).
- The Schmitt trigger input is activated.
- The weak pull-up and pull-down resistors are activated or not depending on the value in the GPIOx_PUPDR register.
- The data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
10.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up and pull-down resistors are disabled by hardware.
- Read access to the input data register gets the value 0.

The figure below shows the high-impedance, analog-input configuration of the I/O port bits.

10.3.13 Using the HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs.

When the HSE or LSE oscillator is switched on (by setting the HSEON or LSEON bit in the RCC_CSR register), the oscillator takes control of its associated pins and the GPIO configuration of these pins has no effect.
When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input, and the OSC.OUT or OSC32_OUT pin can still be used as normal GPIO.

10.3.14 **Using the GPIO pins in the RTC supply domain**

The PC13/PC14/PC15 GPIO functionality is lost when the core supply domain is powered off (when the device enters Standby mode). In this case, if their GPIO configuration is not bypassed by the RTC configuration, these pins are set in an analog input mode.

For details about I/O control by the RTC, refer to **Section 49.3: RTC functional description**

10.3.15 **Privileged and unprivileged modes**

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

10.3.16 **High-speed low-voltage mode (HSLV)**

The I/Os can increase their maximum speed at low voltage by configuring them in HSLV mode. The I/O HSLV bit controls whether the I/O output speed is optimized to operate at 3.3 V (default setting) or at 1.8 V (HSLV = 1). The CSI must be enabled in order to configure the I/Os in HSLV mode.

Procedure to set the HSLV mode for the XSPIx interface:
1. Set the user option byte for the targeted interface (XSPI1_HSLV, XSPI2_HSLV) in the **FLASH option byte word 1 status register (FLASH_OBW1SR)**.
2. Enable the target interface EN_XSPIM1, EN_XSPIM2 in the **PWR control register 2 (PWR_CSR2)**.
3. Activate the HSLV on the targeted i/f (XSPI1_IOHSLV, XSPI2_IOHSLV) in the **SBS I/O compensation cell control and status register (SBS_CCCSR)**.

Procedure to set the HSLV mode for the FMC interface:
1. Set the user option byte for the GPIO and the shared XSPI interface (VDDIO_HSLV, XSPI2_HSLV) in the **FLASH option byte word 1 status register (FLASH_OBW1SR)**.
2. Enable the target interface EN_XSPIM2 in the **PWR control register 2 (PWR_CSR2)**.
3. Activate the HSLV on the targeted i/f (IOHSLV, XSPI2_IOHSLV) in the **SBS I/O compensation cell control and status register (SBS_CCCSR)**.

**Caution:** The I/O HSLV configuration bit must not be set if the I/O supply (VDD) is above 2.7 V. Setting it while the voltage is higher than 2.7 V can damage the device.

**Caution:** The GPIOs are all programed with the same HSLV setting, except those from dedicated power rail (OCTO, HEXA and USB):
- XSPIM2 rail: PN[0:12]
- XSPIM1 rail: PO[0:5], PP[0:15]
- USB no software compensation setting

GPIOs that are shared with the USB are limited in terms of frequency, to hundreds of kHz for UCPD-based GPIOs, and up to 1 MHz for USBHS and USBFS-based GPIOs. Refer to the datasheet for the exact values. In order to be used the IO supply must be above 2.7 V for the USBHS/FS.
Note: When using an FMC with HSLV both the XSPIM2 and GPIO must be set to HSLV and on the same supply, still respecting the HSLV I/O supply limitation.

10.3.17 I/O compensation cell

The I/O commutation slew rate (fall / rise) can be adapted by software depending on process, voltage and temperature conditions, in order to reduce the I/O noise on the power supply.

10.4 GPIO registers

This section gives a detailed description of the GPIO registers.

The peripheral registers can be written in word, half word or byte mode.

10.4.1 GPIO port mode register (GPIOx_MODER) (x = A to G, M to P)

Address offset: 0x00
Reset value: 0xABFF FFFF (port A)
Reset value: 0xFFFF FEBF (port B)
Reset value: 0xFFFF FFFF (ports C...H and M...P)

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>MODEy[1:0]: Port x configuration I/O pin y (y = 15 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
### 10.4.2 GPIO port output type register (GPIOx_OTYPER) (x = A to G, M to P)

Address offset: 0x04  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OT31:16</td>
<td>Res.</td>
</tr>
<tr>
<td>30</td>
<td>OT30</td>
<td>Res.</td>
</tr>
<tr>
<td>29</td>
<td>OT29</td>
<td>Res.</td>
</tr>
<tr>
<td>28</td>
<td>OT28</td>
<td>Res.</td>
</tr>
<tr>
<td>27</td>
<td>OT27</td>
<td>Res.</td>
</tr>
<tr>
<td>26</td>
<td>OT26</td>
<td>Res.</td>
</tr>
<tr>
<td>25</td>
<td>OT25</td>
<td>Res.</td>
</tr>
<tr>
<td>24</td>
<td>OT24</td>
<td>Res.</td>
</tr>
<tr>
<td>23</td>
<td>OT23</td>
<td>Res.</td>
</tr>
<tr>
<td>22</td>
<td>OT22</td>
<td>Res.</td>
</tr>
<tr>
<td>21</td>
<td>OT21</td>
<td>Res.</td>
</tr>
<tr>
<td>20</td>
<td>OT20</td>
<td>Res.</td>
</tr>
<tr>
<td>19</td>
<td>OT19</td>
<td>Res.</td>
</tr>
<tr>
<td>18</td>
<td>OT18</td>
<td>Res.</td>
</tr>
<tr>
<td>17</td>
<td>OT17</td>
<td>Res.</td>
</tr>
<tr>
<td>16</td>
<td>OT16</td>
<td>Res.</td>
</tr>
</tbody>
</table>

Bits 31:16: Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)  
These bits are written by software to configure the I/O output type.
- 0: Output push-pull (reset state)
- 1: Output open-drain

**Note:** The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

### 10.4.3 GPIO port output speed register (GPIOx_OSPEEDR) (x = A to G, M to P)

Address offset: 0x08  
Reset value: 0x0C00 0000 (for port A)  
Reset value: 0x0000 00C0 (for port B)  
Reset value: 0x0000 0000 (for the other ports)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OSPEED15:10</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>OSPEED14:9</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>OSPEED13:8</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>OSPEED12:7</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>OSPEED11:6</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>OSPEED10:5</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>OSPEED9:4</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>OSPEED8:3</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>OSPEED7:2</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>OSPEED6:1</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>OSPEED5:0</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **OSPEEDy:** Port x configuration I/O pin y (y = 15 to 0)  
These bits are written by software to configure the I/O output speed.
- 00: Low speed
- 01: Medium speed
- 10: High speed
- 11: Very-high speed

**Note:** Refer to the device datasheet for the frequency specifications and the power supply and load conditions for each speed.  
The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
10.4.4 GPIO port pull-up/pull-down register (GPIOx_PUPDR) (x = A to G, M to P)

Address offset: 0x0C
Reset value: 0x6400 0000 (for port A)
Reset value: 0x0000 0100 (for port B)
Reset value: 0x0000 0000 (for the other ports)

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>PUPDy[1:0]: Port x configuration I/O pin y (y = 15 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are written by software to configure the I/O pull-up or pull-down</td>
</tr>
<tr>
<td>00:</td>
<td>No pull-up, pull-down</td>
</tr>
<tr>
<td>01:</td>
<td>Pull-up</td>
</tr>
<tr>
<td>10:</td>
<td>Pull-down</td>
</tr>
<tr>
<td>11:</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

10.4.5 GPIO port input data register (GPIOx_IDR) (x = A to G, M to P)

Address offset: 0x10
Reset value: 0x0000 XXXX

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

Bits 15:0 IDy: Port x input data I/O pin y (y = 15 to 0)
These bits are read-only. They contain the input value of the corresponding I/O port.

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
10.4.6 GPIO port output data register (GPIOx_ODR) (x = A to G, M to P)

Address offset: 0x14
Reset value: 0x0000 0000

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 ODy: Port output data I/O pin y (y = 15 to 0)
These bits can be read and written by software.

Note: For atomic bit set/reset, the OD bits can be individually set and/or reset by writing to the GPIOx_BSRR or GPIOx_BRR registers (x = A to G) (x = M to P).
The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

10.4.7 GPIO port bit set/reset register (GPIOx_BSRR) (x = A to G, M to P)

Address offset: 0x18
Reset value: 0x0000 0000

Bits 31:16 BRy: Port x reset I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit
1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.
The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Bits 15:0 BSy: Port x set I/O pin y (y = 15 to 0)
These bits are write-only. A read to these bits returns the value 0x0000.

0: No action on the corresponding ODy bit
1: Sets the corresponding ODy bit

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
10.4.8 GPIO port configuration lock register (GPIOx_LCKR) (x = A to G, M to P)

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the configuration of the GPIO. During the write sequence, the value of LCKR[15:0] must not change. When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset.

Note: A specific write sequence is used to write to the GPIOx_LCKR register. Only word access (32-bit long) is allowed during this locking sequence.

Each lock bit freezes a specific configuration register (control and alternate function registers).

Address offset: 0x1C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:17 Reserved, must be kept at reset value.</th>
<th>Bit 16 LCKK: Lock key</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit can be read any time. It can only be modified using the lock key write sequence.</td>
<td>This bit can be read any time. It can only be modified using the lock key write sequence.</td>
</tr>
<tr>
<td>0: Port configuration lock key not active</td>
<td>0: Port configuration lock key not active</td>
</tr>
<tr>
<td>1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.</td>
<td>1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.</td>
</tr>
<tr>
<td>- LOCK key write sequence:</td>
<td>- LOCK key write sequence:</td>
</tr>
<tr>
<td>- LOCK key read</td>
<td>- LOCK key read</td>
</tr>
<tr>
<td>RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)</td>
<td>RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)</td>
</tr>
</tbody>
</table>

Note: During the LOCK key write sequence, the value of LCK[15:0] must not change.

Any error in the lock sequence aborts the LOCK.

After the first LOCK sequence on any bit of the port, any read access on the LCKK bit returns 1 until the next MCU reset or peripheral reset.

Bits 15:0 LCKy: Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when the LCKK bit is 0

0: Port configuration not locked
1: Port configuration locked

Note: The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
### 10.4.9 GPIO alternate function low register (GPIOx_AFRL) (x = A to G, M to P)

Address offset: 0x20  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>AFSELy[3:0]: Alternate function selection for port x I/O pin y (y = 7 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are written by software to configure alternate function I/Os.</td>
</tr>
<tr>
<td></td>
<td>0000: AF0</td>
</tr>
<tr>
<td></td>
<td>0001: AF1</td>
</tr>
<tr>
<td></td>
<td>0010: AF2</td>
</tr>
<tr>
<td></td>
<td>0011: AF3</td>
</tr>
<tr>
<td></td>
<td>0100: AF4</td>
</tr>
<tr>
<td></td>
<td>0101: AF5</td>
</tr>
<tr>
<td></td>
<td>0110: AF6</td>
</tr>
<tr>
<td></td>
<td>0111: AF7</td>
</tr>
<tr>
<td></td>
<td>1000: AF8</td>
</tr>
<tr>
<td></td>
<td>1001: AF9</td>
</tr>
<tr>
<td></td>
<td>1010: AF10</td>
</tr>
<tr>
<td></td>
<td>1011: AF11</td>
</tr>
<tr>
<td></td>
<td>1100: AF12</td>
</tr>
<tr>
<td></td>
<td>1101: AF13</td>
</tr>
<tr>
<td></td>
<td>1110: AF14</td>
</tr>
<tr>
<td></td>
<td>1111: AF15</td>
</tr>
</tbody>
</table>

**Note:** The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
10.4.10 GPIO alternate function high register (GPIOx_AFRH) (x = A to G, M to P)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **AFSELy[3:0]**: Alternate function selection for port x I/O pin y (y = 15 to 8)

These bits are written by software to configure alternate function I/Os.

- 0000: AF0
- 0001: AF1
- 0010: AF2
- 0011: AF3
- 0100: AF4
- 0101: AF5
- 0110: AF6
- 0111: AF7
- 1000: AF8
- 1001: AF9
- 1010: AF10
- 1011: AF11
- 1100: AF12
- 1101: AF13
- 1110: AF14
- 1111: AF15

*Note: The bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.*
### 10.4.11 GPIO port bit reset register (GPIOx_BRR) (x = A to G, M to P)

- **Address offset:** 0x28
- **Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BR15</td>
<td>BR14</td>
<td>BR13</td>
<td>BR12</td>
<td>BR11</td>
<td>BR10</td>
<td>BR9</td>
<td>BR8</td>
<td>BR7</td>
<td>BR6</td>
<td>BR5</td>
<td>BR4</td>
<td>BR3</td>
<td>BR2</td>
<td>BR1</td>
<td>BR0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**Bits 31:16**  Reserved, must be kept at reset value.

**Bits 15:0**  **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns the value 0x0000.

- 0: No action on the corresponding ODy bit
- 1: Reset the corresponding ODy bit

**Note:** The bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.
### 10.4.12 GPIO register map

#### Table 86. GPIO register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>GPIOx_MODER</td>
<td>0x00</td>
<td>GPIOx_OTYPER</td>
<td>0x00</td>
<td>GPIOx_BSRR</td>
<td>0x00</td>
<td>GPIOx_ODR</td>
<td>0x00</td>
<td>GPIOx_PUPDR</td>
<td>0x00</td>
<td>GPIOx_LCKR</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x04</td>
<td>GPIOx_OTYPER</td>
<td>0x04</td>
<td>GPIOx_OTYPER</td>
<td>0x04</td>
<td>GPIOx_OTYPER</td>
<td>0x04</td>
<td>GPIOx_OTYPER</td>
<td>0x04</td>
<td>GPIOx_OTYPER</td>
<td>0x04</td>
<td>GPIOx_OTYPER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x08</td>
<td>GPIOx_OTYPER</td>
<td>0x08</td>
<td>GPIOx_OTYPER</td>
<td>0x08</td>
<td>GPIOx_OTYPER</td>
<td>0x08</td>
<td>GPIOx_OTYPER</td>
<td>0x08</td>
<td>GPIOx_OTYPER</td>
<td>0x08</td>
<td>GPIOx_OTYPER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0C</td>
<td>GPIOx_OTYPER</td>
<td>0x0C</td>
<td>GPIOx_OTYPER</td>
<td>0x0C</td>
<td>GPIOx_OTYPER</td>
<td>0x0C</td>
<td>GPIOx_OTYPER</td>
<td>0x0C</td>
<td>GPIOx_OTYPER</td>
<td>0x0C</td>
<td>GPIOx_OTYPER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x10</td>
<td>GPIOx_MODER</td>
<td>0x10</td>
<td>GPIOx_MODER</td>
<td>0x10</td>
<td>GPIOx_MODER</td>
<td>0x10</td>
<td>GPIOx_MODER</td>
<td>0x10</td>
<td>GPIOx_MODER</td>
<td>0x10</td>
<td>GPIOx_MODER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x14</td>
<td>GPIOx_MODER</td>
<td>0x14</td>
<td>GPIOx_MODER</td>
<td>0x14</td>
<td>GPIOx_MODER</td>
<td>0x14</td>
<td>GPIOx_MODER</td>
<td>0x14</td>
<td>GPIOx_MODER</td>
<td>0x14</td>
<td>GPIOx_MODER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x18</td>
<td>GPIOx_MODER</td>
<td>0x18</td>
<td>GPIOx_MODER</td>
<td>0x18</td>
<td>GPIOx_MODER</td>
<td>0x18</td>
<td>GPIOx_MODER</td>
<td>0x18</td>
<td>GPIOx_MODER</td>
<td>0x18</td>
<td>GPIOx_MODER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x1C</td>
<td>GPIOx_MODER</td>
<td>0x1C</td>
<td>GPIOx_MODER</td>
<td>0x1C</td>
<td>GPIOx_MODER</td>
<td>0x1C</td>
<td>GPIOx_MODER</td>
<td>0x1C</td>
<td>GPIOx_MODER</td>
<td>0x1C</td>
<td>GPIOx_MODER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x20</td>
<td>GPIOx_MODER</td>
<td>0x20</td>
<td>GPIOx_MODER</td>
<td>0x20</td>
<td>GPIOx_MODER</td>
<td>0x20</td>
<td>GPIOx_MODER</td>
<td>0x20</td>
<td>GPIOx_MODER</td>
<td>0x20</td>
<td>GPIOx_MODER</td>
</tr>
<tr>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
<td></td>
<td>(x = A to IJ)</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>
Table 86. GPIO register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x24   | GPIOx_AFRH    | AFS15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS14 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS13 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS12 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS11 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS09  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | AFS08  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x28   | GPIOx_BRR     | BR15 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR14  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR13  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR12  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR11  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR10  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR09  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR08  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR07  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR06  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR05  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR04  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR03  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR02  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR01  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|        |               | BR00  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Refer to Section 2.3 for the register boundary addresses.
11 Peripheral interconnect

11.1 Introduction

Several peripherals have direct connections between them. This allows autonomous communication and or synchronization between peripherals, saving CPU resources, thus power supply consumption.

In addition, these hardware connections remove software latency and allow design of predictable system.

Depending on peripherals, these interconnections can operate in various power modes: Run, Sleep or Stop modes.

11.2 Connection summary

Table 87. Peripheral interconnect matrix (1) (2)

| Source   | TIM1 | TIM2 | TIM3 | TIM4 | TIM5 | TIM6 | TIM7 | TIM8 | TIM9 | TIM10 | TIM11 | TIM12 | TIM13 | TIM14 | TIM15 | TIM16 | TIM17 | LPTIM1 | LPTIM2 | LPTIM3 | LPTIM4 | LPTIM5 | ADC1/2 | ADI | SAI1/2 | GP/HPDMA | IRTIM | TAMP | RTC | AES/AES |
|----------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|-------|----|--------|----------|-------|------|-----|--------|
| TIM1     | 1 1 1 1 1 | - - 1 1 | - - | - - | - - | - - | - - | - - | - - | 2 | - | 11 | 2 | - | 11 | - | - | - | - | - | - | - | - |
| TIM2     | 1 | - 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | - 2 | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM3     | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM4     | 1 1 1 1 | - 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM5     | 1 1 1 1 | - 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM6     | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM7     | - - | - | - - | - - | - - | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM9     | 1 1 | 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM12    | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM13    | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM14    | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM15    | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM16    | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 11 | - | 10 | - | - | - | - | - | - | - | - | - | - | - | - | - |
| TIM17    | 1 1 1 1 | 1 1 1 | - - | - 1 1 | - - | - - | - - | - - | 2 | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| LPTIM1   | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| LPTIM2   | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| LPTIM3   | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| LPTIM4   | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| LPTIM5   | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| ADC1/2   | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| ADF      | - - | - - | - - | - - | - - | - - | - - | - - | - - | - | 11 | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
Table 87. Peripheral interconnect matrix \(^{(1)}\) \(^{(2)}\) (continued)

| Source                | TIM1 | TIM2 | TIM3 | TIM4 | TIM5 | TIM6 | TIM7 | TIM8 | TIM9 | TIM10 | TIM11 | TIM12 | TIM13 | TIM14 | TIM15 | TIM16 | TIM17 | LPT1M1 | LPT1M2 | LPT1M3 | LPT1M4 | LPT1M5 | LPT1M6 | LPT1M7 | ADC1/2 | ADF | SA11/2 | GPIHPDMA | IRTIM | TAM | RTC | AES/SAES |
|-----------------------|------|------|------|------|------|------|------|------|------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|--------|--------|--------|--------|-------|-------|---------|-------|-----|-----|---------|
| GPDMA                 | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | 11    | -     | -     | -     | -     | -     | -     | 11     | -      | 11     | 11     | 11     | 11     | -      | -     | -     | -       |       |     |     |         |
| HPDMA                 | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | 11    | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| U(S)ART               | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | 11    | -     | -     | -     | -      | 5      | 5      | 5      | 5      | -      | 5     | -     | -       |       |     |     |         |
| LPUART                | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| HSE                   | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | 11    | -     | -     | -     | -      | 5      | 5      | 5      | 5      | -      | 5     | -     | -       |       |     |     |         |
| LSE / LSI             | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | 5     | 5     | 5     | 5     | 5     | 5     | -      | -      | 11     | 11     | 11     | 11     | -      | -     | -     | -       |       |     |     |         |
| SPI                   | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | 11     | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| EXTI                  | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | 2      | 4      | 11     | -      | -      | -     | -     | -       |       |     |     |         |
| TAM                   | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | 11    | -     | -     | 11    | -     | -     | 5      | 5      | 5      | 5      | -      | -     | 11    | -       |       |     |     |         |
| RTC Alarm             | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | 6     | 6     | -     | -      | -      | -      | 15     | 17     | -      | 17     | -      | -     | -       |       |     |     |         |
| VBAT and Temp monitoring | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | 11     | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| VCORE, VBAT           | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | 12    | -     | -     | -     | -     | -     | -      | -      | -      | 12     | -     | -     | -       |       |     |     |         |
| VREFINT               | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| T sensor              | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| CSS in LSE            | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | 16    | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| System error          | 9    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | 9     | 9     | 9     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| AES/SAES              | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | 17    | -     | -     | -     | -      | -      | -      | 17     | -      | -      | -      | -     | -     | -       |       |     |     |         |
| System Flash          | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| SAI1/2                | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | 11    | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| DCMIPP                | 14   | 14   | 14   | 14   | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | 11     | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| DMA2D                 | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | 11     | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| JPEG                  | -    | -    | -    | -    | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| LTDC                  | 14   | 14   | 14   | 14   | -    | -    | -    | -    | -    | -     | -     | -     | -     | -     | -     | -     | -     | -     | -      | 11     | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| GFXTIM1               | 8    | -    | 8    | -    | 8    | -    | -    | -    | -    | -     | -     | -     | -     | -     | 11    | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |
| USB OTG               | 8    | -    | 8    | -    | 8    | -    | -    | -    | -    | -     | -     | -     | -     | -     | 11    | -     | -     | -     | -      | -      | -      | -      | -      | -      | -     | -     | -       |       |     |     |         |

1. Letters in the table correspond to the type of connection described in Section 11.2: Connection summary
2. The "-" symbol in a gray cell means no interconnect.
11.3 Interconnection details

11.3.1 Master to slave interconnection for timers

From timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM9/TIM12/TIM13/TIM14/TIM15/TIM16/TIM17) to timer (TIM1/TIM2/TIM3/TIM4/TIM5/TIM9/TIM12/TIM13/TIM14/TIM15/TIM16/TIM17)

Purpose

Some of the TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in master mode, it can reset, start, stop or clock the counter of another timer configured in slave mode.

Timer synchronization

The synchronization modes are detailed in:
- Section 41.3.30: Timer synchronization for advanced-control timers TIM1
- Section 42.4.23: Timer synchronization for general-purpose timers TIM2/TIM3/TIM4/TIM5
- Section 44.4.20: Timer synchronization (TIM9/TIM12 only) for the general-purpose timer (Lite no DT) TIM9/TIM12
- Section 45.4.26: Timer synchronization (TIM15 only) for the general-purpose timer (Lite) TIM15

Triggering signals

The output (from master) is on signal TIMx_TRGO (and TIMx_TRGO2 for TIM1) following a configurable timer event. It can be also from signals tim16_oc1 and tim17_oc1 in case of TIM16/TIM17. The input (to slave) is on signals TIMx_ITR0/ITR1/ITR2/ITR3.

The possible master/slave connections are given in:
- Table 408: Internal trigger connection for advanced-control timers TIM1
- Table 431: TIMx internal trigger connection for the general-purpose timers TIM2/TIM3/TIM4/TIM5
- Table 452: TIMx internal trigger connection for general-purpose timers TIM9/TIM12
- Table 464: TIMx internal trigger connection for general-purpose timers TIM15

Active power mode

Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:
- Table 420: Effect of low-power modes on TIM1
- Table 445: Effect of low-power modes on TIM6/TIM7
- Table 438: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5
- Table 454: Effect of low-power modes on TIM9/TIM12/TIM13/TIM14
- Table 470: Effect of low-power modes on TIM15/TIM16/TIM17
11.3.2 **Triggers to ADCs**

From EXTI, timers (TIM1/TIM2/TIM3/TIM4/TIM6/TIM9/TIM12/TIM15) and LP timers (LPTIM1/ LPTIM2/LPTIM3) to ADC1

**Purpose**

A conversion, or a sequence of conversions, can be triggered either by software or by an external event (such as timer capture or input pins). For ADC1, if the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events can trigger a conversion with the selected polarity. More details in:

- *Section 27.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*
- *ADC configuration register (ADC_CFGR)*
- *ADC injected sequence register (ADC_JSQR)*

General-purpose timers (TIM2/TIM3/TIM4), basic timer (TIM6), advanced-control timers (TIM1) and general-purpose timer (TIM9/TIM12/TIM15) can be used to generate the ADC triggering event through the timer outputs tim_oc and tim_trgo.

Low-power timers (LPTIM1/ LPTIM2/LPTIM3) can be used to generate the ADC triggering event through the LPTIM channels (TIMx synchronization described in *Section 41.3.30: Timer synchronization* for TIM1) in addition to the EXTI on channels 11 and 15.

**Triggering signals**

For ADC1, the input triggering signals and the description of the interconnection between ADC1, and timers, are given in:

- *Table 229: ADC interconnection*
- *Section 27.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN,JEXTSEL, JEXTEN)*
- *Section 27.4.25: Timing diagrams example (single/continuous modes, hardware/software triggers)*

**Active power mode**

This interconnection is active in Run and Sleep modes for all ADCs. The timers are active in Run and Sleep mode only. The effects of low-power modes are given in:

- *Table 420: Effect of low-power modes on TIM1*
- *Table 438: Effect of low-power modes on TIM2/TIM3/TIM4/TIM5*
- *Table 454: Effect of low-power modes on TIM9/TIM12/TIM13/TIM14*
- *Table 476: STM32H7Rx/7Sx LPTIM features*
11.3.3 ADC analog watchdogs as triggers to timers

From ADC1/ADC2 to TIM1

**Purpose**

The internal analog watchdog output signals coming from ADC1, are connected to on-chip timers. ADC1 can provide trigger event through analog watchdog signals to advanced-control timers (TIM1) in order to reset, start, stop or clock the counter.

Settings description of the ADC analog watchdog and timer trigger, are provided in:

- *Section 41.3.6: External trigger input*
- *Table 409: Interconnect to the tim_etr input multiplexer* for the internal ADC1 sources connected to TIM1 (tim_etr) input multiplexer
- *Table 228: ADC internal input/output signals* for the ADC_AWDx

**Triggering signals**

The output (from ADC) is on signals ADCn_AWDx_OUT, with n being the ADC instance and x = 1, 2, 3 (three watchdogs per ADC). The input (to timer) is on signal TIMx_ETR (external trigger).

**Active power mode**

ADC1 are active in Run and Sleep modes.

11.3.4 Triggers on ADF1

From EXTI to ADF1

**Purpose**

EXTI (EXTI15) can be used to generate a triggering event on ADF1 module and start an A/C conversion.

A description is given in:

- *Section 27.4.2: ADC pins and internal signals*
- *Section 30.4.9: Start-up sequence examples*

**Triggering signals**

The adf_trgi trigger input is the triggering input signal. The ADF trigger inputs connections are detailed in:

- *Table 257: ADF trigger connections*

**Active power mode**

This interconnection remains active down to Stop (SVOS high) for ADF1, assuming the trigger source remains active.

11.3.5 Internal input and clock sources to timers

From HSE, LSE, HSI CSI and MCO to TIM (TIM9, TIM12), LP timers (LPTIM1/LPTIM2/LPTIM3/LPTIM4/LPTIM5) and RTC
Purpose

A timer input or timer counter can receive different clock sources and can be used to calibrate internal oscillator on a reference clock for example.

Internal clocks (CSI, HSI) and microcontroller output clock (MCO) can be used as input to timers:

- HSI and CSI are assigned to general purpose timers TIM12 as external inputs signals. This allows the calibration. HSI/CSI can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in). Inputs assignment and clock selection description are detailed in:
  - Section 44.4.5: Clock selection for TIM12
  - External clock mode1: Table 442: TIM internal input/output signals for TIM12, tim_ti1_in2 (HSI) and tim_ti1_in3(CSI)
- Microcontroller output clock (MCO) is connected as external input to general-purpose timers TIM9. This allows the calibration or clock measurement. When the low-speed external (LSE) oscillator is used, no additional hardware connections are required. This feature is given in Table 450: Interconnect to the tim_ti1 input multiplexer
- LSI and LSE can be selected as input capture 2 to LPTIM1 as described in Table 485: LPTIM1 input capture 2 connection
- HSI/1024 CSI/128 can be selected as input capture 2 to LPTIM2 as described in Table 486: LPTIM2 input capture 2 connection.

Triggering signals

lptim_ic2_mux1 LPTIM input capture selection can be set in the LPTIM configuration register 2 (LPTIM_CFGR2). For timers, the internal clock signal can be selected as counter clock provided by an external clock source in mode1 (tim_ti1_in) and mode2 (external trigger input tim_etr_in).

Active power mode

This feature is available under Run and Sleep modes.

11.3.6 Triggers to low-power timers

From EXTI, TAMP and RTC alarm to LP timers (LPTIM1/LPTIM2/LPTIM3/LPTIM4/LPTIM5)

Purpose

LPTIM1/LPTIM2/LPTIM3/LPTIM4/LPTIM5 counters may be started either by software or after the detection of an active edge on one of the eight trigger inputs.

GPIO can also be selected as LPTIM input capture selection or LPTIM input selection, according to the LPTIM configuration register 2 (LPTIM_CFGR2).

Triggering signals

This trigger feature is described in Table 46.4.7: Trigger multiplexer and the following sections. The input selection is described in Table 481: LPTIM1/2/3/4/5 external trigger connection.

Active power mode

This interconnection remains active down to Stop mode.
11.3.7 RTC wakeup as inputs to timers
From RTC to timers (TIM16)

**Purpose**
RTC wakeup interrupt can be used as input to general-purpose timers (TIM16) channel 1.

**Triggering signals**
RTC wakeup signal is connected to tim_ti1_in3 signal as described in *Table 450: Interconnect to the tim_ti1 input multiplexer* for TIM15.

**Active power mode**
This interconnection is active down to Stop. Timers are not active but the count is performed at wakeup.

11.3.8 USB OTG SOF as trigger to timers
From USB OTG SOF to TIM2

**Purpose**
USB OTG SOF (start-of-frame) can generate a trigger to the general-purpose timer TIM2/TIM5. The USB connection to TIM2/TIM5 are described in *Table 431: TIMx internal trigger connection*.

**Triggering signals**
The tim_itr13 internal signal is generated by OTG HS SOF, the tim_itr14 internal signal is generated by OTG FS SOF.

**Active power mode**
This interconnection is active in Run and Sleep modes.

11.3.9 System errors as break signals to timers
The following system-level errors can trigger a timer break event: Cortex-M7 lockup, PVD low threshold, CSS, double ECC errors in AXISRAM1, AXISRAM3, ITCM, DTCM, backup RAM and flash memory.

The purpose of the break event is to disable the timer outputs to protect the power transistors they control.

System errors apply in RUN and Sleep modes.

**Active power mode**
Timers are optionally active in Run and Sleep modes. The effects of low-power modes on TIMx are given in:
- *Table 420: Effect of low-power modes on TIM1*
- *Table 470: Effect of low-power modes on TIM15/TIM16/TIM17*
11.3.10 **Triggers to GPDMA/HPDMA**

From EXTI, RTC (wakeup), timers (TIM1/TIM2/TIM3/TIM4/TIM5/TIM6/TIM7/TIM9/TIM12/TIM15), LP timers (LPTIM1/LPTIM3/LPTIM4/LPTIM5), HP/GPDMA1 transfer complete (gpdma1_chx_tcf/lhdma1_chx_tcf), SPI6, LPUART,

**Purpose**

A HP/GPDMA trigger can be assigned to a HP/GPDMA channel x. A programmed HP/GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event. The trigger mode can also be programmed to condition the LLI link transfer. More details are given in the sections below:

- Section 12.3.6: GPDMA triggers and Section 13.3.6: HPDMA triggers
- Section 12.4.12: GPDMA triggered transfer and Section 13.4.12: HPDMA triggered transfer
- Section 12.8.9: GPDMA channel x transfer register 2 (GPDMA_CxTR2) and Section 13.8.9: HPDMA channel x transfer register 2 (HPDMA_CxTR2) for more details on:
  - Trigger selection TRIGSEL[5:0] field
  - Trigger mode (LLI) defined by TRIGM[1:0].
  - Trigger polarity as defined by TRIGPOL[1:0]

**Triggering signals**

GPDMA trigger mapping is specified in Table 94: Programmed GPDMA1 trigger, according to GPDMA_CxTR2.TRIGSEL[5:0].

HPDMA trigger mapping is specified in Table 106: Programmed HPDMA1 trigger, according to HPDMA_CxTR2.TRIGSEL[5:0].

**Active power mode**

Assuming sources are active down to Stop modes, this interconnection remains functional in Stop.

Refer to:
- Section 12.6: GPDMA in low-power modes
- Section 13.6: HPDMA in low-power modes

11.3.11 **Internal analog signals to analog peripherals**

From internal analog source to ADC (ADC1)

**Purpose**

The internal reference voltage (VREFINT), the internal temperature sensor (VSENSE) monitoring channel are connected to ADC (ADC1) input channels. In addition, the internal
digital core voltage (VCORE) and VBAT are connected to input channels. This is according to:

- Section 27.4.32: Temperature sensor
- Section 27.4.33: VBAT supply monitoring
- Section 27.4.34: Monitoring the internal voltage reference
- Section 27.4.35: Monitoring the supply voltage

Active power mode
These interconnections remain in Stop modes if the selected peripheral is kept active. Refer to Section 27.5: ADC in low-power modes

11.3.12 ADC data filtering by the ADF1
From ADC1 to ADF1

Purpose
The ADF1 allows the connection of up to two ADCs to the filter path. For each filter, the DATSRC[1:0] field in the ADF digital filter configuration register x (ADF_DFLTxCICR) is used to select either data from the ADCs in:

- Section 27.4.4: ADC connectivity
- Table 229: ADC interconnection
  - mdf_adcitf1_dat[15:0] to adc_da
- Table 268: Effect of low-power modes on ADF

Active power mode
This feature remains available down to Sleep mode.

11.3.13 Internal tamper sources
From internal peripherals, clocks or monitoring to tamper.

Purpose
In order to detect any abnormal activity or tentative to corrupt the device, tampers are introduced and alert the system of such undesired event. Different actions can be taken in consequences.

List of tamper sources can be found in Table 21: Internal tampers in TAMP

Active power mode
This interconnection is active in all power modes if the tamper source is activated.
11.3.14 Output from tamper to RTC

From TAMP to RTC

Purpose

The RTC can timestamp a tamper event in order to retrieve history in time of such detection. The RTC can also control GPIOs and set a signal based on tamp or alarm status outside the MCU.

Refer to Section 49.3.3: GPIOs controlled by the RTC and TAMP for more details.

Active power mode

This interconnection remain active in all power modes.

11.3.15 Encryption keys to AES/SAES

From TAMP backup registers, system Flash memory to and in between SAES and AES

Purpose

The encryption mechanism requires an hardware key that must be stored in a protected non-volatile memory. Different approaches are implemented in order to load them in a non-readable way. Tamper backup registers or system Flash can be used to store respectively BHK or RHUK, and to implement a dedicated bus to pass it to the SAES.

Refer to Section 36.4.14: SAES operation with wrapped keys for more details.

The AES encryption mechanism (faster than the SAES) can benefit from the sharing key of the SAES. Refer to Section 36.4.15: SAES operation with shared keys for more details.

Active power mode

AES and SAES are operating under Run and Sleep modes.
12 General purpose direct memory access controller (GPDMA)

12.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

12.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during low-power modes (see Section 12.3.2)
  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see Section 12.3.1)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
- Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers
- 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see Section 12.3.1)
- Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
- Programmable GPDMA request and trigger selection
- Programmable GPDMA half transfer and transfer complete events generation
- Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers

- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

### 12.3 GPDMA implementation

#### 12.3.1 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

<table>
<thead>
<tr>
<th>Channel x</th>
<th>Hardware parameters</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dma_fifo_size[x]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>dma_addressing[x]</td>
<td></td>
</tr>
<tr>
<td>x = 0 to 11</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>x = 12 to 15</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Channel x (x = 0 to 11) is implemented with:
- a FIFO of 8 bytes, 2 words
- fixed/contiguously incremented addressing

These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and AHB SRAM.

Channel x (x = 12 to 15) is implemented with:
- a FIFO of 32 bytes, 8 words
- 2D addressing

These channels may be also used for GPDMA transfers, between a demanding AHB peripheral and AHB SRAM, or for transfers from/to AXI SRAM or external AXI memories.
12.3.2 GPDMA autonomous mode in low-power modes

The GPDMA autonomous mode and wake-up feature are implemented in the device low-power modes as per the table below.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake-up</td>
<td>GPDMA1 in Sleep mode</td>
</tr>
</tbody>
</table>

12.3.3 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[6:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.REQSEL[6:0]</th>
<th>Selected GPDMA request</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>adc1_dma</td>
</tr>
<tr>
<td>1</td>
<td>adc2_dma</td>
</tr>
<tr>
<td>2</td>
<td>crypt_in_dma</td>
</tr>
<tr>
<td>3</td>
<td>crypt_out_dma</td>
</tr>
<tr>
<td>4</td>
<td>saes_out_dma</td>
</tr>
<tr>
<td>5</td>
<td>saes_in_dma</td>
</tr>
<tr>
<td>6</td>
<td>hash_dma</td>
</tr>
<tr>
<td>7</td>
<td>tim1_cc_dma</td>
</tr>
<tr>
<td>8</td>
<td>tim1_cc_dma</td>
</tr>
<tr>
<td>9</td>
<td>tim1_cc_dma</td>
</tr>
<tr>
<td>10</td>
<td>tim1_cc_dma</td>
</tr>
<tr>
<td>11</td>
<td>tim1_upd_dma</td>
</tr>
<tr>
<td>12</td>
<td>tim1_trg_dma</td>
</tr>
<tr>
<td>13</td>
<td>tim1_com_dma</td>
</tr>
<tr>
<td>14</td>
<td>tim2_cc_dma</td>
</tr>
<tr>
<td>15</td>
<td>tim2_cc_dma</td>
</tr>
<tr>
<td>16</td>
<td>tim2_cc_dma</td>
</tr>
<tr>
<td>17</td>
<td>tim2_cc_dma</td>
</tr>
<tr>
<td>18</td>
<td>tim2_upd_dma</td>
</tr>
<tr>
<td>19</td>
<td>tim2_trg_dma</td>
</tr>
<tr>
<td>20</td>
<td>tim3_cc_dma</td>
</tr>
<tr>
<td>21</td>
<td>tim3_cc_dma</td>
</tr>
<tr>
<td>22</td>
<td>tim3_cc_dma</td>
</tr>
</tbody>
</table>
Table 90. Programmed GPDMA1 request (continued)

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.REQSEL[6:0]</th>
<th>Selected GPDMA request</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>tim3_cc_dma</td>
</tr>
<tr>
<td>24</td>
<td>tim3_upd_dma</td>
</tr>
<tr>
<td>25</td>
<td>tim3_trg_dma</td>
</tr>
<tr>
<td>26</td>
<td>tim4_cc_dma</td>
</tr>
<tr>
<td>27</td>
<td>tim4_cc_dma</td>
</tr>
<tr>
<td>28</td>
<td>tim4_cc_dma</td>
</tr>
<tr>
<td>29</td>
<td>tim4_cc_dma</td>
</tr>
<tr>
<td>30</td>
<td>tim4_upd_dma</td>
</tr>
<tr>
<td>31</td>
<td>tim4_trg_dma</td>
</tr>
<tr>
<td>32</td>
<td>tim5_cc_dma</td>
</tr>
<tr>
<td>33</td>
<td>tim5_cc_dma</td>
</tr>
<tr>
<td>34</td>
<td>tim5_cc_dma</td>
</tr>
<tr>
<td>35</td>
<td>tim5_cc_dma</td>
</tr>
<tr>
<td>36</td>
<td>tim5_upd_dma</td>
</tr>
<tr>
<td>37</td>
<td>tim5_trg_dma</td>
</tr>
<tr>
<td>38</td>
<td>tim6_upd_dma</td>
</tr>
<tr>
<td>39</td>
<td>tim7_upd_dma</td>
</tr>
<tr>
<td>40</td>
<td>tim15_cc_dma</td>
</tr>
<tr>
<td>41</td>
<td>tim15_cc_dma</td>
</tr>
<tr>
<td>42</td>
<td>tim15_upd_dma</td>
</tr>
<tr>
<td>43</td>
<td>tim15_trg_dma</td>
</tr>
<tr>
<td>44</td>
<td>tim15_com_dma</td>
</tr>
<tr>
<td>45</td>
<td>tim16_cc_dma</td>
</tr>
<tr>
<td>46</td>
<td>tim16_upd_dma</td>
</tr>
<tr>
<td>47</td>
<td>tim16_com_dma</td>
</tr>
<tr>
<td>48</td>
<td>tim17_cc_dma</td>
</tr>
<tr>
<td>49</td>
<td>tim17_upd_dma</td>
</tr>
<tr>
<td>50</td>
<td>tim17_com_dma</td>
</tr>
<tr>
<td>51</td>
<td>spi1_rx_dma</td>
</tr>
<tr>
<td>52</td>
<td>spi1_tx_dma</td>
</tr>
<tr>
<td>53</td>
<td>spi2_rx_dma</td>
</tr>
<tr>
<td>54</td>
<td>spi2_tx_dma</td>
</tr>
<tr>
<td>55</td>
<td>spi3_rx_dma</td>
</tr>
<tr>
<td>56</td>
<td>spi3_tx_dma</td>
</tr>
<tr>
<td>57</td>
<td>spi4_rx_dma</td>
</tr>
<tr>
<td>GPDMA_CxTR2.REQSEL[6:0]</td>
<td>Selected GPDMA request</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>58</td>
<td>spi4_tx_dma</td>
</tr>
<tr>
<td>59</td>
<td>spi5_rx_dma</td>
</tr>
<tr>
<td>60</td>
<td>spi5_tx_dma</td>
</tr>
<tr>
<td>61</td>
<td>spi6_rx_dma</td>
</tr>
<tr>
<td>62</td>
<td>spi6_tx_dma</td>
</tr>
<tr>
<td>63</td>
<td>sai1_a_dma</td>
</tr>
<tr>
<td>64</td>
<td>sai1_b_dma</td>
</tr>
<tr>
<td>65</td>
<td>sai2_a_dma</td>
</tr>
<tr>
<td>66</td>
<td>sai2_b_dma</td>
</tr>
<tr>
<td>67</td>
<td>i2c1_rx_dma</td>
</tr>
<tr>
<td>68</td>
<td>i2c1_tx_dma</td>
</tr>
<tr>
<td>69</td>
<td>i2c2_rx_dma</td>
</tr>
<tr>
<td>70</td>
<td>i2c2_tx_dma</td>
</tr>
<tr>
<td>71</td>
<td>i2c3_rx_dma</td>
</tr>
<tr>
<td>72</td>
<td>i2c3_tx_dma</td>
</tr>
<tr>
<td>73</td>
<td>usart1_rx_dma</td>
</tr>
<tr>
<td>74</td>
<td>usart1_tx_dma</td>
</tr>
<tr>
<td>75</td>
<td>usart2_rx_dma</td>
</tr>
<tr>
<td>76</td>
<td>usart2_tx_dma</td>
</tr>
<tr>
<td>77</td>
<td>usart3_rx_dma</td>
</tr>
<tr>
<td>78</td>
<td>usart3_tx_dma</td>
</tr>
<tr>
<td>79</td>
<td>uart4_rx_dma</td>
</tr>
<tr>
<td>80</td>
<td>uart4_tx_dma</td>
</tr>
<tr>
<td>81</td>
<td>uart5_rx_dma</td>
</tr>
<tr>
<td>82</td>
<td>uart5_tx_dma</td>
</tr>
<tr>
<td>83</td>
<td>uart7_rx_dma</td>
</tr>
<tr>
<td>84</td>
<td>uart7_tx_dma</td>
</tr>
<tr>
<td>85</td>
<td>uart8_rx_dma</td>
</tr>
<tr>
<td>86</td>
<td>uart8_tx_dma</td>
</tr>
<tr>
<td>87</td>
<td>cordic_rd_dma</td>
</tr>
<tr>
<td>88</td>
<td>cordic_wr_dma</td>
</tr>
<tr>
<td>89</td>
<td>lptim1_ic1_dma</td>
</tr>
<tr>
<td>90</td>
<td>lptim1_ic2_dma</td>
</tr>
<tr>
<td>91</td>
<td>lptim1_ue_dma</td>
</tr>
<tr>
<td>92</td>
<td>lptim2_ic1_dma</td>
</tr>
</tbody>
</table>
### 12.3.4 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

#### Table 91. Programmed GPDMA1 request as a block request

<table>
<thead>
<tr>
<th>GPDMA block requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim1_ue_dma</td>
</tr>
<tr>
<td>lptim2_ue_dma</td>
</tr>
<tr>
<td>lptim3_ue_dma</td>
</tr>
</tbody>
</table>
### 12.3.5 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of
the data transfer from the peripheral which does also support this feature.

<table>
<thead>
<tr>
<th>GPDMA channel x with peripheral early termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 0 and x = 15</td>
</tr>
</tbody>
</table>

This GPDMA support is activated when the channel x is programmed with
GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data
transfer completion, before that the GPDMA has transferred the whole block
(see Section 12.4.14 for more details).

### 12.3.6 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[5:0]
in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the
selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.TRIGSEL[5:0]</th>
<th>Selected GPDMA trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>hpdma1_ch0_tc</td>
</tr>
<tr>
<td>1</td>
<td>hpdma1_ch1_tc</td>
</tr>
<tr>
<td>2</td>
<td>hpdma1_ch2_tc</td>
</tr>
<tr>
<td>3</td>
<td>hpdma1_ch3_tc</td>
</tr>
<tr>
<td>4</td>
<td>hpdma1_ch4_tc</td>
</tr>
<tr>
<td>5</td>
<td>hpdma1_ch5_tc</td>
</tr>
<tr>
<td>6</td>
<td>hpdma1_ch6_tc</td>
</tr>
<tr>
<td>7</td>
<td>hpdma1_ch7_tc</td>
</tr>
<tr>
<td>8</td>
<td>hpdma1_ch8_tc</td>
</tr>
<tr>
<td>9</td>
<td>hpdma1_ch9_tc</td>
</tr>
<tr>
<td>10</td>
<td>hpdma1_ch10_tc</td>
</tr>
<tr>
<td>11</td>
<td>hpdma1_ch11_tc</td>
</tr>
<tr>
<td>12</td>
<td>hpdma1_ch12_tc</td>
</tr>
<tr>
<td>13</td>
<td>hpdma1_ch13_tc</td>
</tr>
<tr>
<td>14</td>
<td>hpdma1_ch14_tc</td>
</tr>
<tr>
<td>15</td>
<td>hpdma1_ch15_tc</td>
</tr>
</tbody>
</table>
### Table 94. Programmed GPDMA1 trigger (continued)

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.TRIGSEL[5:0]</th>
<th>Selected GPDMA trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>lptim1_ch1</td>
</tr>
<tr>
<td>17</td>
<td>lptim1_ch2</td>
</tr>
<tr>
<td>18</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>19</td>
<td>lptim2_ch2</td>
</tr>
<tr>
<td>20</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>21</td>
<td>lptim3_ch2</td>
</tr>
<tr>
<td>22</td>
<td>lptim4_out</td>
</tr>
<tr>
<td>23</td>
<td>lptim5_out</td>
</tr>
<tr>
<td>24</td>
<td>exti0</td>
</tr>
<tr>
<td>25</td>
<td>rtc_wut_trg</td>
</tr>
<tr>
<td>26</td>
<td>lpuart1_wkup</td>
</tr>
<tr>
<td>27</td>
<td>reserved</td>
</tr>
<tr>
<td>28</td>
<td>spi6_it</td>
</tr>
<tr>
<td>29</td>
<td>gpdma1_ch0_tc</td>
</tr>
<tr>
<td>30</td>
<td>gpdma1_ch1_tc</td>
</tr>
<tr>
<td>31</td>
<td>gpdma1_ch2_tc</td>
</tr>
<tr>
<td>32</td>
<td>gpdma1_ch3_tc</td>
</tr>
<tr>
<td>33</td>
<td>gpdma1_ch4_tc</td>
</tr>
<tr>
<td>34</td>
<td>gpdma1_ch5_tc</td>
</tr>
<tr>
<td>35</td>
<td>gpdma1_ch6_tc</td>
</tr>
<tr>
<td>36</td>
<td>gpdma1_ch7_tc</td>
</tr>
<tr>
<td>37</td>
<td>gpdma1_ch8_tc</td>
</tr>
<tr>
<td>38</td>
<td>gpdma1_ch9_tc</td>
</tr>
<tr>
<td>39</td>
<td>gpdma1_ch10_tc</td>
</tr>
<tr>
<td>40</td>
<td>gpdma1_ch11_tc</td>
</tr>
<tr>
<td>41</td>
<td>gpdma1_ch12_tc</td>
</tr>
<tr>
<td>42</td>
<td>gpdma1_ch13_tc</td>
</tr>
<tr>
<td>43</td>
<td>gpdma1_ch14_tc</td>
</tr>
<tr>
<td>44</td>
<td>gpdma1_ch15_tc</td>
</tr>
<tr>
<td>45</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>46</td>
<td>tim1_trgo2</td>
</tr>
<tr>
<td>47</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>48</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>49</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>50</td>
<td>tim5_trgo</td>
</tr>
</tbody>
</table>
12.4 GPDMA functional description

12.4.1 GPDMA block diagram

![GPDMA Block Diagram](image)

Table 94. Programmed GPDMA1 trigger (continued)

<table>
<thead>
<tr>
<th>GPDMA_CxTR2.TRIGSEL[5:0]</th>
<th>Selected GPDMA trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>tim6_trgo</td>
</tr>
<tr>
<td>52</td>
<td>tim7_trgo</td>
</tr>
<tr>
<td>53</td>
<td>tim9_trgo</td>
</tr>
<tr>
<td>54</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>55</td>
<td>tim15_trgo</td>
</tr>
</tbody>
</table>

(1) Refer to the device implementation table for the number of channels.
12.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 88. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)
12.4.3 GPDMA channel suspend and resume

The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note that an ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 89. GPDMA channel suspend and resume sequence

Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

12.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the
channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.

3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.

4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.

5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

**Figure 90. GPDMA channel abort and restart sequence**

12.4.5 **GPDMA linked-list data structure**

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.
The base address in memory of the data structure of a next LLI_{n+1} of a channel \(x\) is the sum of the following:

- the link base address of the channel \(x\) (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLI_{n} of the channel \(x\).

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the UT1, UT2, UB1, USA, UDA and ULL bits, plus UB2 and UT3, in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3 or GPDMA_CxBR2) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

**Static linked-list data structure**

For example, when the update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel \(x\) (\(x = 0\) to \(11\)) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see Figure 91) and including the first linked-list register file (LLI_{0}) and the next LLIs (such as LLI_{1}, LLI_{2}) in the memory
- channel \(x\) (\(x = 12\) to \(15\)), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2 (see Figure 92), and including the first linked-list register file (LLI_{0}) and the next LLIs (such as LLI_{1}, LLI_{2}) in the memory
Figure 91. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x
Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see Figure 93).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UDA = ULL = 1, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxDAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in Figure 94.
The user must program GPDMA_CxLLR for each LLI to be 32-bit aligned and not to exceed the 64-Kbyte addressable space pointed by GPDMA_CxLBAR.

12.4.6 Linked-list item transfer execution

A LLI transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI+1, as defined by the GPDMA_CxLLR value of the LLI.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMACxBR1.BNDT[15:0] = 0 and GPDMACxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCxCR.LSM = 1), as described in the next sections.
12.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI₀, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLIₙ₋₁, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLIN-1
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLIN-1 transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:
- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:
- If the last LLIN is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLIN is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see Section 12.4.14 for more details).

In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 95 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 95 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 95, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 12.3.5).
Figure 95. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)

Channel state = Idle

- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active

- Valid user setting?
  - Yes
    - BNDT ≠ 0?
      - Yes
        - Executing once the data transfer from the register file
      - No
        - No transfer error?
          - Yes
            - Loading next LLI into the register file
          - No
            - LLR ≠ 0?
              - Yes
                - Setting DTEF = 1 Disabling DMA channel
              - No
                - Setting USEF = 1 Disabling DMA channel
              - No transfer error?
                - Yes
                  - Setting USEF = 1 Disabling DMA channel
                - No
                  - Valid user setting?
                    - Yes
                      - Setting TCF = 1 Disabling DMA channel
                    - No
                      - Setting USEF = 1 Disabling DMA channel

- Setting USEF = 1 Disabling DMA channel
- Setting USEF = 1 Disabling DMA channel
- Setting USEF = 1 Disabling DMA channel
- Setting USEF = 1 Disabling DMA channel
- Setting USEF = 1 Disabling DMA channel
- Setting USEF = 1 Disabling DMA channel

End
Run-time inserting a LLIₙ via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLIₙ₋₁ (start of the LLIₙ loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLIₙ₋₁:

- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see Section 12.3.6 for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLIₙ in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLIₙ₋₁ of the channel x, the software can pause the primary channel x after its LLIₙ₋₁ data transfer, until it is indeed written the LLIₙ.

The figure below depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLIₙ₋₁ data transfer that does not need a trigger. The triggering mode of this LLIₙ₋₁ is used to load the next LLIₙ.
12.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 12 to 15), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.
- GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
- GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
- GPDMA_CxTR3 for channel x (x = 12 to 15) defines the source/destination additional address offset between burst transfers.
- GPDMA_CxBR2 for channel x (x = 12 to 15) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
- GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when \( GPDMA_CxBR1.BRC[10:0] = 0 \), and \( GPDMA_CxBR1.BNDT[15:0] = 0 \))
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

**Note:** If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated by the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see Section 12.4.14 for more details).

The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_{n}.LLR.LA = LLI_{n-1}.LLR.LA + constant).

**Figure 97** depicts the GPDMA channel execution mode, and its programming in link step mode.

**Note:** **Figure 97** is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In **Figure 97**, BNDT \( \neq 0 \) is the typical condition for starting the first data transfer. This condition becomes (BNDT \( \neq 0 \) and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 12.3.5).
Figure 97. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)

Channel state = Idle
- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active
- Valid user setting?
  - Y: Setting USEF = 1
  - N: Disabling DMA channel
- BNDT ≠ 0?
  - Y: Executing once the data transfer from the register file
  - N: Setting DTEF = 1
- No transfer error?
  - Y: Setting ULEF = 1
  - N: Disabling DMA channel
- LLR ≠ 0?
  - Y: Loading next LLI into the register file
  - N: Setting USEF = 1
- No transfer error?
  - Y: Valid user setting?
    - Y: Setting TCF = 1
    - N: Disabling DMA channel
  - N: Disabling DMA channel
- Valid user setting?
  - Y: Setting USEF = 1
  - N: Disabling DMA channel

End
Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_{n} from the memory, as shown in the figure below.

**Figure 98. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode**

Run-time replacing a LLI_{n} with a new LLI_{n'} in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_{n'}, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_{n} from the memory by overwriting directly the linked-list register file with the new LLI_{n'}, as shown in the figure below.
Figure 99. Replace with a new LLI_{n'} in register file in link step mode

Run-time replacing a LLI_{n} with a new LLI_{n'} in link step mode (in the memory)

The software can build and insert a new LLI_{n'} and LLI_{n'+1}' in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_{n} from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new LLI_{n'}) as shown in the figure below.
Figure 100. Replace with a new LLI_{n} and LLI_{n+1}' in memory in link step mode (option 1)

LSM = 1 with 1-stage linked-list programming:
Overwriting the (pre)loaded LLI_{n} linked-list register file with a new LLI_{n}' and LLI_{n+1}' in memory and
overwrite partly linked-list register file
(DMA_CxBR1.BNDT = 0 and DMA_CxLLR to point to new LLI_{n}')
DMA executes LLI_{n-1} and load LLI_{n} then CPU builds (LLI_{n}' and LLI_{n+1}') and overwrite (BR1 and LLR)

DMA Channel

✅ Executing LLI_{n-1} data transfer

Loading LLI_{n}

Transfer complete interrupt

CPU

Build LLI_{n}' and LLI_{n+1}' in memory

Write DMA_CxBR1.BNDT = 0
Write DMA_CxLLR to point to new LLI_{n}'

Enable DMA channel

✅ Loading LLI_{n}'

Transfer complete interrupt

✅ Executing LLI_{n+1}' data transfer

Loading LLI_{n+1}'

Transfer complete interrupt
Run-time replacing a \( LLIn \) with a new \( LLIn' \) in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the \( LLIn-1 \) and loads a formerly elaborated \( LLIn \) from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new \( LLIn' \) and a new \( LLIn_{n+1}' \).
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for \( LLIn_{n-1} \) is then restricted to overwrite \( GPDMA\_CxBR1.BNDT[15:0] = 0 \) and \( GPDMA\_CxLLR \) to point to new \( LLIn' \), as shown in the figure below.

Figure 101. Replace with a new \( LLIn' \) and \( LLIn_{n+1}' \) in memory in link step mode (option 2)

- LSM = 1 with 1-stage linked-list programming:
  - Overwriting the (pre)loaded \( LLIn \), linked-list register file by building new \( LLIn' \) and \( LLIn_{n+1}' \) in memory while disabling (temporary) channel interrupt at NVIC level, and overwriting \( DMA\_CxBR1.BNDT = 0 \) and \( DMA\_CxLLR \) to point to new \( LLIn' \).
  - DMA executes \( LLIn_{n-1} \) and loading \( LLIn \) while CPU builds \( (LLIn' \) and \( LLIn_{n+1}' \)), then CPU overwrites \( (BR1 \) and \( LLR) \).
12.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLIn+1 in the memory to finally complete the channel by this LLIn+1 (clear the GPDMA_CxLLR of this LLIn+1), before that this LLIn+1 is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see Section 12.4.4.)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLIn-1.

*Figure 102* shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

**Note:** *Figure 102 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.*

In *Figure 102*, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 12.3.5).
Figure 102. GPDMA channel execution and linked-list programming

Channel state = Idle

- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active

- Valid user setting?
  - Yes
  - No
    - BNDT ≠ 0?
      - Yes
      - No
        - Setting USEF = 1
          - Disabling DMA channel
    - No
      - Setting TCF = 1
        - Disabling DMA channel
      - Executing once the data transfer from the register file
      - No transfer error?
        - Yes
        - N
          - LLR ≠ 0?
            - Yes
            - N
              - Setting ULEF = 1
                - Disabling DMA channel
            - Loading next LLI into the register file
            - No transfer error?
              - Yes
              - N
                - Setting USEF = 1
                  - Disabling DMA channel
              - Valid user setting?
                - Yes
                - N
                  - LSM = 1?
                    - Yes
                    - N
                      - Setting USEF = 1
                        - Disabling DMA channel
                    - Setting TCF = 1
                      - Disabling DMA channel
                  - No
                    - Setting USEF = 1
                      - Disabling DMA channel
            - Setting DTEF = 1
              - Disabling DMA channel
      - Setting USEF = 1
        - Disabling DMA channel

End
12.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see Section 12.3.1 for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

<table>
<thead>
<tr>
<th>Table 95. Programmed GPDMA source/destination burst</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDW_LOG2[1:0]</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>01</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
</tbody>
</table>

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.
GPDMA burst with 2D addressing (channel x = 12 to 15)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by the figure below with a 2D destination buffer.
Programmable address jumps 1) after burst and 2) after block.
Example:
burst: \( I \times \text{words} \) (DBL=1; DDW_LOG2=\( \text{b}10 \))
block: \( J \times \text{bursts} \) (BNDT=\( J^*4 \))
LLI: \( K \times \text{blocks} \) (BRC=K-1)
**GPDMA FIFO-based burst**

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

**GPDMA source burst**

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

\[
\text{when } \text{FIFO level } \leq 2^{\text{dma_fifo_size}[x]} - (SBL_1[5:0]+1) \times 2^{\text{SDW_LOG2}[1:0]}
\]

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- \(2^{\text{dma_fifo_size}[x]}\) is the half of the FIFO size of the channel x, in bytes (see *Section 12.3.1* for the implementation details and dma_fifo_size[x] value).
- \((SBL_1[5:0]+1) \times 2^{\text{SDW_LOG2}[1:0]}\) is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIOR[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

**GPDMA destination burst**

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

\[
\text{when } \text{FIFO level } \geq (DBL_1[5:0]+1) \times 2^{\text{DDW_LOG2}[1:0]}
\]

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- \((DBL_1[5:0]+1) \times 2^{\text{DDW_LOG2}[1:0]}\) is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

**GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size**

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.
- If the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

**Note:** In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

**GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment**

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see Table 96).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. **Byte-based source reordering**
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.

2. **Data width conversion by packing, unpacking, padding or truncation,** if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.

3. **Byte-based destination re-ordering:**
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

**Note:** Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.
The table below lists the possible data handling from the source to the destination.

**Table 96. Programmed data handling**

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(1)</th>
<th>SB X</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0][2]</th>
<th>DB X</th>
<th>DH X</th>
<th>Destination data stream(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Byte</td>
<td>x</td>
<td>00</td>
<td>Byte</td>
<td>xx</td>
<td>x</td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, 0P)</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0B3,0B2,0B1,0B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B30,B20,B10,B00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (RA, SE)</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>SB3,SB2,SB1,SB0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B3S,B2S,B1S,B0S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (PACK)</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>B7B6,B7B4,B3B2,B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B7B6,B7B4,B3B2,B1B0</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>x</td>
<td>01</td>
<td>Half-word</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>000B1,000B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, 0P)</td>
<td>1</td>
<td></td>
<td>0</td>
<td></td>
<td>00B1,000B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>0B1,000,B000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (RA, SE)</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>SSBB1,SSBB0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>SBB1,SSBB0S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>SB1,SS,SB0SS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B1,SS,SB1SS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (PACK)</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>B7B6B5B4,B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B7B6B5B4,B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>B5B4B7B6,B1B0B2B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B5B4B7B6,B1B0B2B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, LT)</td>
<td>01 (LA, RT)</td>
<td>x</td>
<td></td>
<td>B6B5,B2,B0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00</td>
<td>01</td>
<td>x</td>
<td></td>
<td></td>
<td>B6B5,B2,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>01</td>
<td></td>
<td>0</td>
<td></td>
<td>B7,B6,B7,B3,B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>B7,B6,B7,B3,B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1,B0</td>
</tr>
</tbody>
</table>

(1) Source data stream, destination data stream 
(2) Source address mode, destination address mode
### Table 96. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(1)</th>
<th>SB X</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM<a href="2">1:0</a></th>
<th>DB X</th>
<th>DH X</th>
<th>Destination data stream(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Half-word</td>
<td>(B_7B_6B_5B_4, B_3B_2, B_1B_0)</td>
<td>x</td>
<td>10</td>
<td>Word</td>
<td>00 (RA, 0P)</td>
<td>0</td>
<td>1</td>
<td>(B_7B_6B_5B_4, B_3B_2B_1B_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>(B_0B_7B_6B_5B_2B_3B_0)</td>
</tr>
<tr>
<td>00</td>
<td>Byte</td>
<td>(B_7B_6B_5B_4, B_3B_2B_1B_0)</td>
<td>0</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)</td>
<td>x</td>
<td></td>
<td>(B_12B_9B_4B_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>(LA, RT)</td>
<td></td>
<td></td>
<td></td>
<td>(B_{15}B_{11}B_7B_3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>(UNPACK)</td>
<td></td>
<td></td>
<td></td>
<td>(B_7B_6B_5B_4B_3B_2B_1B_0)</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>(B_7B_6B_5B_4, B_3B_2B_1B_0)</td>
<td>0</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)</td>
<td>x</td>
<td></td>
<td>(B_3B_2B_0B_1B_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td>(LA, RT)</td>
<td></td>
<td></td>
<td></td>
<td>(B_7B_6B_3B_2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x</td>
<td>(UNPACK)</td>
<td></td>
<td></td>
<td></td>
<td>(B_6B_7B_2B_3)</td>
</tr>
</tbody>
</table>

**Notes:**
(1) Stream 1
(2) \(PAM = 00\) results in a zero word

**Data Handling Modes:**
- **SDW** (Source Data Word)
- **SBX** (Source Byte Select)
- **DDW** (Destination Data Word)
- **PAM** (Programmed Access Mode)
- **DBX** (Destination Byte Select)
- **DHX** (Destination Half-Word Select)
12.4.11 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory-mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[6:0] field (see Section 12.3.3). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

1. Data stream is timely ordered starting from the byte with the lowest index (B0).
2. RA = right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.
Caution:  The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[6:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRI0[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRI0[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to-memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRI0[1:0]). The arbitration policy, as illustrated in Figure 104, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels
  
  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.

- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2
  
  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.

Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.

Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

**Figure 104. GPDMA arbitration policy**

GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as BQ3) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (Breq0) is equal to the allocated bandwidth to req1 (Breq1).
    \[ B_{req0} = B_{req1} = \frac{1}{2} \times (1 - B_{Q3}) \]
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (BP0) is 3 times less than the allocated bandwidth to req1 (BP1).  
    \[ B_{req0} = B_{P0} = \frac{1}{2} \times \frac{1}{2} \times (1 - B_{Q3}) = \frac{1}{4} \times (1 - B_{Q3}) \]
    \[ B_{req1} = B_{P1} = \frac{1}{2} + 1 \times \frac{1}{2} \times (1 - B_{Q3}) = \frac{3}{4} \times (1 - B_{Q3}) \]
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (BP0) is 5 times less than the allocated bandwidth to req1 (BP2).  
    \[ B_{req0} = B_{P0} = \frac{1}{2} \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{1}{6} \times (1 - B_{Q3}) \]
    \[ B_{req1} = B_{P2} = \frac{1}{2} + 1 + 1 \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{5}{6} \times (1 - B_{Q3}) \]

The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
Example 2: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:

- The allocated bandwidth to the set of requests of priority 0 (B_P0) is
  \[ B_{P0} = \frac{1}{3} \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{1}{9} \times (1 - B_{Q3}) \]

- The allocated bandwidth to the set of requests of priority 1 (B_P1) is
  \[ B_{P1} = \left(\frac{1}{3} + \frac{1}{2}\right) \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{5}{18} \times (1 - B_{Q3}) \]

- The allocated bandwidth to the set of requests of priority 2 (B_P2) is
  \[ B_{P2} = \left(\frac{1}{3} + \frac{1}{2} + 1\right) \times \frac{1}{3} \times (1 - B_{Q3}) = \frac{11}{18} \times (1 - B_{Q3}) \]

- The allocated bandwidth to any request n (B_n) among the N requests of that priority P_i (i = 0 to 2) is
  \[ B_n = \frac{1}{N} \times B_{P_i} \]

- The allocated bandwidth to any request n of priority 0 (B_n, P_0) is
  \[ B_{n, P0} = \frac{1}{N} \times \frac{1}{9} \times (1 - B_{Q3}) \]

  \[ B_{n, P1} = \frac{1}{N} \times \frac{5}{18} \times (1 - B_{Q3}) \]

  \[ B_{n, P2} = \frac{1}{N} \times \frac{11}{18} \times (1 - B_{Q3}) \]

In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I \geq 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.

- J requests (J \geq 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.

- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1

- L requests (L \geq 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- \[ B_l = B_{Q3} / L \text{ for } L > 0 \text{ (else: } B_l = 0) \]

The bandwidth for each non-time sensitive queue is:

- B_{Q0} = \frac{1}{3} \times (1 - B_{Q3})
- B_{Q1} = \frac{1}{3} \times (1 - B_{Q3})
- B_{Q2} = \frac{1}{3} \times (1 - B_{Q3})

The bandwidth for the set of requests with priority 0 is:

- B_{P0} = I / (I + J + K) \times B_{Q0}

The bandwidth for each request i with priority 0 is:

- B_i = B_{P0} / I \text{ for } L > 0 \text{ (else } B_{P0} = 0)\]

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- B_{P1,Q0} = J / (I + J + K) \times B_{Q0}
The bandwidth for the set of requests with priority 1 and routed to queue 1 is:
• $B_{P1,Q1} = J / (J + K) \cdot B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:
• $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request $j$ with priority 1 is:
• $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:
• $B_{P2,Q0} = K / (I + J + K) \cdot B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:
• $B_{P2,Q1} = K / (J + K) \cdot B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:
• $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:
• $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request $k$ with priority 2 is:
• $B_k = B_{P2} / K$ (for $K > 0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any $i$, $j$, $k$ non-time sensitive request are:
• in the general case (when there is at least one request $k$ with a priority 2 ($K > 0$)):
  - $B_i = 1/l * 1/3 * l/(l + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(l + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(l + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
• in the specific case (when there is no request $k$ with a priority 2 ($K = 0$)):
  - $B_i = 1/l * 1/2 * l/(l + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(l + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

### 12.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[5:0] (see Section 12.3.6 for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel $x$ ($x = 12$ to $15$), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).
Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next LLI\textsubscript{n+1}, that updates the GPDMA\_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI\textsubscript{n} trigger.

**Caution:** After a first new trigger hit\textsubscript{n+1} is memorized, if another trigger hit\textsubscript{n+2} is detected and if the hit\textsubscript{n} triggered transfer is still not completed, hit\textsubscript{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA\_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA\_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

**Figure 105. Trigger hit, memorization and overrun waveform**

Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

12.4.13 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half transfer and complete transfer
events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

The figure below illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

**Figure 106. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel**

Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI\textsubscript{N-1} in memory is needed to configure the first block transfer. And the last LLI\textsubscript{N} restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).
The figure below illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

**Figure 107. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel**

### 12.4.14 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see Section 12.3.5).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.

- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see GPDMA channel x transfer register 2 (GPDMA_CxTR2)). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).
In peripheral flow-control mode:
- a destination peripheral with a hardware requested transfer is not supported:
  memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source
  (peripheral) burst size.

12.4.15 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a
privileged agent via GPDMA_PRIVCFGFR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following
access control rules are applied:
- An unprivileged read access to a register field of this channel is forced to return 0,
  except for GPDMA_PRIVCFGFR and GPDMA_RCFGLOCKR that are readable by an
  unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and
destination data transfers are privileged (respectively unprivileged) transfers over the AHB
master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode,
the loading of the next linked-list data structure from the GPDMA memory into its register
file, is automatically performed with privileged (respectively unprivileged) transfers, via the
GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGFR, to keep the
other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to
an unprivileged state, the privileged software must abort the channel or wait until that the
privileged channel is completed before switching. This is needed to dynamically re-allocate
a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so,
and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the
GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x
(GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

12.4.16 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending
on the root cause.

Data transfer error

on a bus access (as a AHB single or a burst) to the source or the destination
- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the
  GPDMA hardware). The channel status register reports an idle state
  (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

**Link transfer error**

on a tentative update of a GPDMA channel register from the programmed LLI in the memory

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

**User setting error**

on a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
– for channel x (x = 12 to 15): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
– a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
– for channel x (x = 12 to 15): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
– for channel x (x = 12 to 15): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
– a programmed double-word source data width (GPDMA_CxTR1.SDW_LOG2[1:0] = 11)
– a programmed double-word destination data width (GPDMA_CxTR1.DDW_LOG2[1:0] = 11)
– a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1.BNDT = 0)

12.4.17 GPDMA autonomous mode

To save dynamic power consumption while the GPDMA executes the programmed linked-list transfers, the GPDMA hardware automatically manages its own clock gating and generates a clock request output signal to the RCC, whenever the device is in Run or low-power modes, provided that the RCC is programmed with the corresponding GPDMA enable control bits.

For more details about the RCC programming, refer to the RCC section of the reference manual.

For mode details about the availability of the GPDMA autonomous feature vs the device low-power modes, refer to Section 12.3.2.

The user can program and schedule the execution of a given GPDMA transfer at a LLI_{n} level of a GPDMA channel x, with GPDMA_CxTR2 as follows:

- The software controls and conditions the input of a transfer with TRIGM[1:0], TRIGPOL[1:0], TRIGSEL[5:0], SWREQ and REQSEL[6:0] for the input trigger and request.
- The software controls and signals the output of a transfer with TCEM[1:0] for generating or not a transfer complete event, and generating or not an associated half data transfer event).

See GPDMA channel x transfer register 2 (GPDMA_CxTR2) for more details.

When used in low-power modes, this functionality enables a CPU wake-up on a specific transfer completion by the enabled GPDMA transfer complete interrupt (GPDMA_CxCR.TCIE = 1) or/and enables to continue with the autonomous GPDMA for operating another LLI_{n+1} transfer over the same channel.

The output channel x transfer complete event, gpdma_chx_tc, can be programmed as a selected input trigger for a channel if this event is looped-back and connected at the
GPDMA level (see Section 12.3.6), allowing autonomous and fine GPDMA inter-channel transfer scheduling, without needing a cleared transfer complete flag (TCF).

A given GPDMA channel x asserts its clock request in one of the following conditions:
- if the next transfer to be executed is programmed as conditioned by a trigger (GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGM[1:0]), only when the trigger hit occurs.
- if the next transfer to be executed is not conditioned by a trigger:
  - if GPDMA_CxTR2.SWREQ = 0, only when the hardware request is asserted by the selected peripheral
  - if GPDMA_CxTR2.SWREQ = 1 (memory-to-memory, GPIO to/from memory), as soon as the GPDMA is enabled

The GPDMA channel x releases its clock request as soon as all the following conditions are met:
- The transfer to be executed is completed.
- The GPDMA channel x is not immediately ready and requested to execute the next transfer.
- If a channel x interrupt was raised, all the flags of the status register that can cause this interrupt, are cleared by a software agent.

When one channel asserts its clock request, the GPDMA asserts its clock request to the RCC. When none channel asserts its clock request, the GPDMA releases its clock request to the RCC.

### 12.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

*Note:* In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

### 12.6 GPDMA in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. GPDMA interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The content of the GPDMA registers is kept when entering Stop mode. The content of the GPDMA registers can be autonomously updated by a next linked-list item from memory, to perform autonomous data transfers. GPDMA interrupts can cause the device to exit Stop mode(^{(1)}).</td>
</tr>
<tr>
<td>Standby</td>
<td>The GPDMA is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Refer to Section 12.3.2 to know if any Stop mode is supported.
12.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 98. GPDMA interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Interrupt enable</th>
<th>Event flag</th>
<th>Event clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPDMA_Channel_x</td>
<td>Transfer complete</td>
<td>GPDMA_CxCR.TCIE</td>
<td>GPDMA_CxSR.TCF</td>
<td>Write 1 to GPDMA_CxFCR.TCF</td>
</tr>
<tr>
<td></td>
<td>Half transfer</td>
<td>GPDMA_CxCR.HTIE</td>
<td>GPDMA_CxSR.HTF</td>
<td>Write 1 to GPDMA_CxFCR.HTF</td>
</tr>
<tr>
<td></td>
<td>Data transfer error</td>
<td>GPDMA_CxCR.DTEIE</td>
<td>GPDMA_CxSR.DTEF</td>
<td>Write 1 to GPDMA_CxFCR.DTEF</td>
</tr>
<tr>
<td></td>
<td>Update link error</td>
<td>GPDMA_CxCR.ULEIE</td>
<td>GPDMA_CxSR.ULEF</td>
<td>Write 1 to GPDMA_CxFCR.ULEF</td>
</tr>
<tr>
<td></td>
<td>User setting error</td>
<td>GPDMA_CxCR.USEIE</td>
<td>GPDMA_CxSR.USEF</td>
<td>Write 1 to GPDMA_CxFCR.USEF</td>
</tr>
<tr>
<td></td>
<td>Suspended</td>
<td>GPDMA_CxCR.SUSPIE</td>
<td>GPDMA_CxSR.SUSPF</td>
<td>Write 1 to GPDMA_CxFCR.SUSPF</td>
</tr>
<tr>
<td></td>
<td>Trigger overrun</td>
<td>GPDMA_CxCR.TOFIE</td>
<td>GPDMA_CxSR.TOF</td>
<td>Write 1 to GPDMA_CxFCR.TOF</td>
</tr>
</tbody>
</table>

A GPDMA channel x event may be:
- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

**Note:** When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:
- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:
- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].
• A half transfer event is an half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].
A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See GPDMA channel x transfer register 2 (GPDMA_CxTR2) for more details.

A transfer error rises in one of the following situations:
• during a single/burst data transfer from the source or to the destination (DTEF)
• during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
• during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)
The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see Section 12.4.16).

A trigger overrun is described in Trigger hit memorization and trigger overrun flag generation.

12.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

12.8.1 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04
Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PRIV15</td>
<td>PRIV14</td>
<td>PRIV13</td>
<td>PRIV12</td>
<td>PRIV11</td>
<td>PRIV10</td>
<td>PRIV9</td>
<td>PRIV8</td>
<td>PRIV7</td>
<td>PRIV6</td>
<td>PRIV5</td>
<td>PRIV4</td>
<td>PRIV3</td>
<td>PRIV2</td>
<td>PRIV1</td>
<td>PRIV0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Res. must be kept at reset value.

Bits 15:0 PRIVx: privileged state of channel x (x = 15 to 0)
0: unprivileged
1: privileged
12.8.2  GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08
Reset value: 0x0000 0000

This register can be written by a software agent with privileged attributes in order to individually lock, for example at boot time, the privileged attributes of any GPDMA channel/resource (to lock the setting of GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK15</td>
<td>LOCK14</td>
<td>LOCK13</td>
<td>LOCK12</td>
<td>LOCK11</td>
<td>LOCK10</td>
<td>LOCK9</td>
<td>LOCK8</td>
<td>LOCK7</td>
<td>LOCK6</td>
<td>LOCK5</td>
<td>LOCK4</td>
<td>LOCK3</td>
<td>LOCK2</td>
<td>LOCK1</td>
<td>LOCK0</td>
</tr>
<tr>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **LOCKx**: lock the configuration of GPDMA_PRIVCFGR.PRIVx, until a global GPDMA reset (x = 15 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.
0: privilege configuration of the channel x is writable.
1: privilege configuration of the channel x is not writable.

12.8.3  GPDMA masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C
Reset value: 0x0000 0000

This is a read register.

This register contains the masked interrupt status bit MISx for each channel x. It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channels, other privileged bit fields returning zero.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIS15</td>
<td>MIS14</td>
<td>MIS13</td>
<td>MIS12</td>
<td>MIS11</td>
<td>MIS10</td>
<td>MIS9</td>
<td>MIS8</td>
<td>MIS7</td>
<td>MIS6</td>
<td>MIS5</td>
<td>MIS4</td>
<td>MIS3</td>
<td>MIS2</td>
<td>MIS1</td>
<td>MIS0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
12.8.4 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register must be written when GPDMA_CxCR.EN = 0.
This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 MISx: masked interrupt status of channel x (x = 15 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

12.8.5 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This is a write register privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 Reserved, must be kept at reset value.
Bit 14 **TOF**: trigger overrun flag clear  
0: no effect  
1: corresponding TOF flag cleared  

Bit 13 **SUSPF**: completed suspension flag clear  
0: no effect  
1: corresponding SUSPF flag cleared  

Bit 12 **USEF**: user setting error flag clear  
0: no effect  
1: corresponding USEF flag cleared  

Bit 11 **ULEF**: update link transfer error flag clear  
0: no effect  
1: corresponding ULEF flag cleared  

Bit 10 **DTEF**: data transfer error flag clear  
0: no effect  
1: corresponding DTEF flag cleared  

Bit 9 **HTF**: half transfer flag clear  
0: no effect  
1: corresponding HTF flag cleared  

Bit 8 **TCF**: transfer complete flag clear  
0: no effect  
1: corresponding TCF flag cleared  

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

### 12.8.6 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 15)  
Reset value: 0x0000 0001  

This is a read register, reporting the channel status.

This register is privileged or non-privileged, depending on the privileged state of the channel (GPDMA_PRIVCFG.RPRIVx).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level  
Number of available write beats in the FIFO, in units of the programmed destination data width (see GPDMA_CxTR1.DDW_LOG2[1:0], in units of bytes, half-words, or words).

**Note:** After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).
Bit 15  Reserved, must be kept at reset value.

Bit 14  **TOF**: trigger overrun flag
        0: no trigger overrun event
        1: a trigger overrun event occurred

Bit 13  **SUSPF**: completed suspension flag
        0: no completed suspension event
        1: a completed suspension event occurred

Bit 12  **USEF**: user setting error flag
        0: no user setting error event
        1: a user setting error event occurred

Bit 11  **ULEF**: update link transfer error flag
        0: no update link transfer error event
        1: a master bus error event occurred while updating a linked-list register from memory

Bit 10  **DTEF**: data transfer error flag
        0: no data transfer error event
        1: a master bus error event occurred on a data transfer

Bit 9   **HTF**: half transfer flag
        0: no half transfer event
        1: a half transfer event occurred
        A half transfer event is either a half block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
        A half block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
        A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8   **TCF**: transfer complete flag
        0: no transfer complete event
        1: a transfer complete event occurred
        A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

Bits 7:1  Reserved, must be kept at reset value.

Bit 0   **IDLEF**: idle flag
        0: channel not in idle state
        1: channel in idle state
        This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
        This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).
12.8.7 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>TOIE: trigger overrun interrupt enable</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>USEIE: use interrupt enable</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>ULEIE: use LLI end interrupt enable</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>DTEIE: data transfer interrupt enable</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>HTIE: hardware trigger interrupt enable</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>TCIE: transfer complete interrupt enable</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserve</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 PRIO[1:0]: priority level of the channel x GPDMA transfer versus others
- 00: low priority, low weight
- 01: low priority, mid weight
- 10: low priority, high weight
- 11: high priority

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 LAP: linked-list allocated port

This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
- 0: port 0 (AHB) allocated
- 1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 LSM: Link step mode

0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed once for the current LLI

First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 TOIE: trigger overrun interrupt enable

0: interrupt disabled
1: interrupt enabled
Bit 13 **SUSPIE**: completed suspension interrupt enable  
  0: interrupt disabled  
  1: interrupt enabled  

Bit 12 **USEIE**: user setting error interrupt enable  
  0: interrupt disabled  
  1: interrupt enabled  

Bit 11 **ULEIE**: update link transfer error interrupt enable  
  0: interrupt disabled  
  1: interrupt enabled  

Bit 10 **DTEIE**: data transfer error interrupt enable  
  0: interrupt disabled  
  1: interrupt enabled  

Bit 9 **HTIE**: half transfer complete interrupt enable  
  0: interrupt disabled  
  1: interrupt enabled  

Bit 8 **TCIE**: transfer complete interrupt enable  
  0: interrupt disabled  
  1: interrupt enabled  

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

Bit 2 **SUSP**: suspend  
  Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:  
  Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).  
  The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in Figure 89.  
  0: write: resume channel, read: channel not suspended  
  1: write: suspend channel, read: channel suspended.  

Bit 1 **RESET**: reset  
  This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).  
  The reset is effective when the channel is in steady state, meaning one of the following:  
  - active channel in suspended state (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = 0 and GPDMA_CxCR.EN = 1)  
  - channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).  
  After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in Figure 90).  
  0: no channel reset  
  1: channel reset
Bit 0  **EN**: enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

### 12.8.8 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of the channel x in GPDMA_PRIVCFG.PRIVx.
This register controls the transfer of a channel x.
This register must be written when GPDMA_CxCR.EN = 0.
This register is read-only when GPDMA_CxCR.EN = 1.
This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.
In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

| Bit 31 | Res. DAP Res. DHX DBX DBL_1[5:0] DINC Res. DDW_LOG2[1:0] |
|--------|----------------|----------------|----------------|----------------|----------------|
| rw     | rw             | rw             | rw             | rw             | rw             |
| 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 |
| Res. SAP SBX PAM[1:0] Res. SBL_1[5:0] SINC Res. SDW_LOG2[1:0] |
| rw     | rw             | rw             | rw             | rw             | rw             |

- **Bit 31**  Reserved, must be kept at reset value.
- **Bit 30**  **DAP**: destination allocated port
  This bit is used to allocate the master port for the destination transfer
  0: port 0 (AHB) allocated
  1: port 1 (AHB) allocated
  *Note*: This bit must be written when EN = 0. This bit is read-only when EN = 1.
- **Bits 29:28**  Reserved, must be kept at reset value.
- **Bit 27**  **DHX**: destination half-word exchange
  If the destination data size is shorter than a word, this bit is ignored.
  If the destination data size is a word:
  0: no halfword-based exchanged within word
  1: the two consecutive (post PAM) half-words are exchanged in each destination word.
Bit 26  **DBX**: destination byte exchange  
If the destination data size is a byte, this bit is ignored.  
If the destination data size is not a byte:  
0: no byte-based exchange within half-word  
1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20  **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63  
The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

**Note:** If a burst transfer crossed a 1-Kbyte address boundary on an AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

**If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.**

Bit 19  **DINC**: destination incrementing burst  
0: fixed burst  
1: contiguously incremented burst  
The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18  Reserved, must be kept at reset value.

Bits 17:16  **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes  
00: byte  
01: half-word (2 bytes)  
10: word (4 bytes)  
11: user setting error reported and no transfer issued

**Note:** Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.  
A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

Bit 15  Reserved, must be kept at reset value.

Bit 14  **SAP**: source allocated port  
This bit is used to allocate the master port for the source transfer  
0: port 0 (AHB) allocated  
1: port 1 (AHB) allocated

**Note:** This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13  **SBX**: source byte exchange within the unaligned half-word of each source word  
If the source data width is shorter than a word, this bit is ignored.  
If the source data width is a word:  
0: no byte-based exchange within the unaligned half-word of each source word  
1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.
Bits 12:11 **PAM[1:0]:** padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

**Condition: PAM_1**
- 00: source data is transferred as right aligned, padded with 0s up to the destination data width
- 01: source data is transferred as right aligned, sign extended up to the destination data width
- 10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

**Condition: PAM_2**
- 00: source data is transferred as right aligned, left-truncated down to the destination data width
- 01: source data is transferred as left-aligned, right-truncated down to the destination data width
- 10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

**Note:** If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]:** source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

**Note:** If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC:** source incrementing burst

- 0: fixed burst
- 1: contiguously incremented burst

The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.
12.8.9 **GPDMA channel x transfer register 2 (GPDMA_CxTR2)**

Address offset: \(0x94 + 0x80 \times x (x = 0 \text{ to } 15)\)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Note: Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.
Bits 31:30  **TCEM[1:0]:** transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

**Note:** If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: channel x (x = 0 to 11), same as 00, channel x (x =12 to 15), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

**Note:** If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x =12 to 15), if any data transfer.

**Note:** If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26  Reserved, must be kept at reset value.

Bits 25:24  **TRIGPOL[1:0]:** trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: no trigger (masked trigger event)
01: trigger on the rising edge
10: trigger on the falling edge
11: same as 00

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:16  **TRIGSEL[5:0]:** trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per Section 12.3.6), with an active trigger event if TRIGPOL[1:0] ≠ 00.
Bits 15:14 **TRIGM[1:0]: trigger mode**

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 12 to 15), for each block if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 11), same as 00; channel x (x = 12 to 15), at 2D/repeated block level. The first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_{n} trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_{n} triggered transfer is still not completed, hit_{n+2} is lost and not memorized.

A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

**Note:** When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.
Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

*Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 12.3.1 for the list of the implemented channels with this feature).*

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see Section 12.3.5 for more details).

**Note:** In peripheral flow control mode, there are the following restrictions:
- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)
- the peripheral must be set as the source of the transfer (DREQ = 0).
- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see Section 12.3.3).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

**Note:** If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[6:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[6:0] is ignored.

Bits 8:7 Reserved, must be kept at reset value.
12.8.10 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

BNDT[15:0]

Bits 31:16  Reserved, must be kept at reset value.
12.8.11 **GPDMA channel x alternate block register 1 (GPDMA_CxBR1)**

Address offset: 0x98 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or non-privileged, depending on the privileged state of channel x (GPDMA_PRIVCFGx.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored to the programmed value (finite/continuous last LLI).

Note: **A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.**

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.
automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI.

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

Bit 31  **BRDDEC**: Block repeat destination address decrement
0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)
1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

**Note:** On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30  **BRSDEC**: Block repeat source address decrement
0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)
1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

**Note:** On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29  **DDEC**: destination address decrement
0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)
1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28  **SDEC**: source address decrement
0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)
1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27  **Reserved:** must be kept at reset value.
Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

- When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.
- Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):
  - If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.
  - If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
  - If all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
  - If GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

- Once the last data transfer is completed (BNDT[15:0] = 0):
  - if GPDMA_CxLLR.UB1 = 1, this field is updated by the LLI in the memory.
  - if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.
  - if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
  - if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

*Note:* A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.
12.8.12 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

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

Bits 31:0 SA[31:0]: source address
This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]
– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].
– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.
12.8.13 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 \textbf{DA}[31:0]: destination address

This field is the pointer to the address from which the next data is written.
During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.
During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]
– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].
– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 12 to 15), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

\textbf{Note}: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.
12.8.14 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 12 to 15)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGER.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

| Bits 31:29 | Reserved, must be kept at reset value. |
| Bits 28:16 | DAO[12:0]: destination address offset increment |
| The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1). |
| Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued. |
| Bits 15:13 | Reserved, must be kept at reset value. |
| Bits 12:0 | SAO[12:0]: source address offset increment |
| The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1). |
| Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued. |
| When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied. |
### 12.8.15 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: \(0xA8 + 0x80 \times x\) (\(x = 12\) to \(15\))

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel \(x\) (GPDMA_PRIVCFGx.PRIVx).

This register controls the transfer of a channel \(x\) at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### BRDAO[15:0]

**Bits 31:16** **BRDAO[15:0]:** Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

**Note:** A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

#### BRSAO[15:0]

**Bits 15:0** **BRSAO[15:0]:** Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

**Note:** A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).
12.8.16  GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

### Bit 31 UT1: Update GPDMA_CxTR1 from memory
- This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
- 0: no GPDMA_CxTR1 update
- 1: GPDMA_CxTR1 update

### Bit 30 UT2: Update GPDMA_CxTR2 from memory
- This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
- 0: no GPDMA_CxTR2 update
- 1: GPDMA_CxTR2 update

### Bit 29 UB1: Update GPDMA_CxBR1 from memory
- This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
- If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
- GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
- 0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
- 1: GPDMA_CxBR1 update

### Bit 28 USA: Update GPDMA_CxSAR from memory
- This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
- 0: no GPDMA_CxSAR update
- 1: GPDMA_CxSAR update
Bit 27  **UDA**: Update GPDMA\_CxDAR register from memory
This bit is used to control the update of GPDMA\_CxDAR from the memory during the link transfer.
0: no GPDMA\_CxDAR update
1: GPDMA\_CxDAR update

Bits 26:17  Reserved, must be kept at reset value.

Bit 16  **ULL**: Update GPDMA\_CxLLR register from memory
This bit is used to control the update of GPDMA\_CxLLR from the memory during the link transfer.
0: no GPDMA\_CxLLR update
1: GPDMA\_CxLLR update

Bits 15:2  **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA\_CxTR1, GPDMA\_CxTR2, GPDMA\_CxBR1, GPDMA\_CxSAR, GPDMA\_CxDAR, and GPDMA\_CxLLR).

Note:  *The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.*

Bits 1:0  Reserved, must be kept at reset value.
## 12.8.17 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFG.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UT1: Update GPDMA_CxTR1 from memory</td>
<td>0: no GPDMA_CxTR1 update</td>
<td>1: GPDMA_CxTR1 update</td>
</tr>
<tr>
<td>30</td>
<td>UT2: Update GPDMA_CxTR2 from memory</td>
<td>0: no GPDMA_CxTR2 update</td>
<td>1: GPDMA_CxTR2 update</td>
</tr>
<tr>
<td>29</td>
<td>UB1: Update GPDMA_CxBR1 from memory</td>
<td>0: no GPDMA_CxBR1 update from memory</td>
<td>1: GPDMA_CxBR1 update from memory</td>
</tr>
<tr>
<td>28</td>
<td>USA: update GPDMA_CxSAR from memory</td>
<td>0: no GPDMA_CxSAR update</td>
<td>1: GPDMA_CxSAR update</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ULL</td>
<td>unfit</td>
</tr>
<tr>
<td>LA</td>
<td>unfit</td>
</tr>
</tbody>
</table>
Bit 27  **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0: no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26  **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25  **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16  **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2  **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

**Note**: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

### 12.8.18 GPDMA register map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| 0x04   | GPDMA_PRIVCFGR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | GPDMA_RCFGLOCKR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | GPDMA_MISR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.
13 High-performance direct memory access controller (HPDMA)

13.1 HPDMA introduction

The high-performance direct memory access (HPDMA) controller is a bus master and system peripheral.

The HPDMA is used to perform programmable data transfers between memory-mapped peripherals, and/or memories via linked-lists, upon the control of an off-loaded CPU.

13.2 HPDMA main features

- Single bidirectional AXI master and single bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral
- Autonomous data transfers during low-power modes (Section 13.3.2)
  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent HPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see Section 13.3.1)
  - Intra-channel HPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel HPDMA transfers chaining via programmable HPDMA input triggers connection to HPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
- Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at block level, between successive burst transfers
- 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see Section 13.3.1)
- Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
- Selection of programmable HPDMA request and trigger
- Generation of programmable HPDMA half-transfer and transfer-complete event
- Pointer to the next linked-list item and its data structure in memory, with automatic update of the HPDMA linked-list control registers
- Channel abort and restart

- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- Privileged/unprivileged support:
  - Support for privileged and unprivileged HPDMA transfers, independently at channel level
  - Privileged-aware AHB slave port

### 13.3 HPDMA implementation

#### 13.3.1 HPDMA channels

A given HPDMA channel \( x \) is implemented with the following features and intended use.

To make the best use of the HPDMA performance, the following table lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested HPDMA transfer.

<table>
<thead>
<tr>
<th>Channel ( x )</th>
<th>Hardware parameters</th>
<th>Features</th>
</tr>
</thead>
</table>
| \( x = 0 \) to \( 11 \) | \( \text{dma}_\text{fifo}_\text{size}[x] \) 3 \( \text{dma}_\text{addressing}[x] \) 0 | Channel \( x \) (\( x = 0 \) to 11) is implemented with:
- a FIFO of 16 bytes, 4 words, 2 double-words
- fixed/contiguously incremented addressing
These channels can be used for HPDMA transfers between an APB or AHB peripheral, an AHB/AXI SRAM, or CPU TCM. |
Table 100. Implementation of HPDMA1 channels (continued)

<table>
<thead>
<tr>
<th>Channel x</th>
<th>dma_fifo_size[x]</th>
<th>dma_addressing[x]</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 12 to 15</td>
<td>5</td>
<td>1</td>
<td>Channel x (x = 12 to 15) is implemented with:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– a FIFO of 64 bytes, 8 double-words</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– 2D addressing</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>These channels can be also used for HPDMA transfers, including AXI external memories.</td>
</tr>
</tbody>
</table>

13.3.2 HPDMA autonomous mode in low-power modes

The HPDMA autonomous mode and wake-up feature are implemented in the device low-power modes as per the table below.

Table 101. HPDMA1 in low-power modes

<table>
<thead>
<tr>
<th>Feature</th>
<th>Low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>wake-up</td>
<td>HPDMA1 in Sleep mode</td>
</tr>
</tbody>
</table>

13.3.3 HPDMA requests

An HPDMA request from a peripheral can be assigned to a HPDMA channel x, via REQSEL[4:0] in HPDMA_CxTR2, provided that SWREQ = 0.

The HPDMA requests mapping is specified in the table below.

Table 102. Programmed HPDMA1 request

<table>
<thead>
<tr>
<th>HPDMA_CxTR2.REQSEL[4:0]</th>
<th>Selected HPDMA request</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>jpeg_rx_dma</td>
</tr>
<tr>
<td>1</td>
<td>jpeg_tx_dma</td>
</tr>
<tr>
<td>2</td>
<td>xspi1_rx_dma</td>
</tr>
<tr>
<td>3</td>
<td>xspi2_rx_dma</td>
</tr>
<tr>
<td>4</td>
<td>spi3_rx_dma</td>
</tr>
<tr>
<td>5</td>
<td>spi3_tx_dma</td>
</tr>
<tr>
<td>6</td>
<td>spi4_rx_dma</td>
</tr>
<tr>
<td>7</td>
<td>spi4_tx_dma</td>
</tr>
<tr>
<td>8</td>
<td>adc1_dma</td>
</tr>
<tr>
<td>9</td>
<td>adc2_dma</td>
</tr>
<tr>
<td>10</td>
<td>adf1_flt0_dma</td>
</tr>
<tr>
<td>11</td>
<td>uart4_rx_dma</td>
</tr>
<tr>
<td>12</td>
<td>uart4_tx_dma</td>
</tr>
</tbody>
</table>
13.3.4 HPDMA block requests

Some HPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in HPDMA_CxTR2 must be set for a correct HPDMA execution of the requested peripheral transfer at the hardware level.

Table 103. Programmed HPDMA1 request as a block request

<table>
<thead>
<tr>
<th>HPDMA block requests</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim2_ue_dma</td>
</tr>
</tbody>
</table>

13.3.5 HPDMA channels with peripheral early termination

An HPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 104. HPDMA channel with peripheral early termination

<table>
<thead>
<tr>
<th>HPDMA channel x with peripheral early termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>x = 15</td>
</tr>
</tbody>
</table>

This HPDMA support is activated when the channel x is programmed with HPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before that the HPDMA has transferred the whole block (see Section 13.4.14 for more details).

Table 105. Programmed HPDMA request with peripheral early termination

<table>
<thead>
<tr>
<th>Programmed HPDMA channel x request with peripheral early termination</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpeg_tx_dma</td>
</tr>
</tbody>
</table>

13.3.6 HPDMA triggers
An HPDMA trigger can be assigned to an HPDMA channel x, via TRIGSEL[5:0] in HPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 106. Programmed HPDMA1 trigger

<table>
<thead>
<tr>
<th>HPDMA_CxTR2.TRIGSEL[5:0]</th>
<th>Selected HPDMA trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>dcmipp_p0frame_evt</td>
</tr>
<tr>
<td>1</td>
<td>dcmip_p0hsync_evt</td>
</tr>
<tr>
<td>2</td>
<td>dcmipp_p0line_evt</td>
</tr>
<tr>
<td>3</td>
<td>dcmipp_p0vsync_evt</td>
</tr>
<tr>
<td>4</td>
<td>dma2d_ctc</td>
</tr>
<tr>
<td>5</td>
<td>dma2d_tc</td>
</tr>
<tr>
<td>6</td>
<td>dma2d_tw</td>
</tr>
<tr>
<td>7</td>
<td>jpeg_eoc_trg</td>
</tr>
<tr>
<td>8</td>
<td>jpeg_ifnf_trg</td>
</tr>
<tr>
<td>9</td>
<td>jpeg_ift_trg</td>
</tr>
<tr>
<td>10</td>
<td>jpeg_ofne_trg</td>
</tr>
<tr>
<td>11</td>
<td>jpeg_oftr_trg</td>
</tr>
<tr>
<td>12</td>
<td>lcd_li</td>
</tr>
<tr>
<td>13</td>
<td>gpu2d_gp_flag0</td>
</tr>
<tr>
<td>14</td>
<td>gpu2d_gp_flag1</td>
</tr>
<tr>
<td>15</td>
<td>gpu2d_gp_flag2</td>
</tr>
<tr>
<td>16</td>
<td>gpu2d_gp_flag3</td>
</tr>
<tr>
<td>17</td>
<td>gfxtim_evt3</td>
</tr>
<tr>
<td>18</td>
<td>gfxtim_evt2</td>
</tr>
<tr>
<td>19</td>
<td>gfxtim_evt1</td>
</tr>
<tr>
<td>20</td>
<td>gfxtim_evt0</td>
</tr>
<tr>
<td>21</td>
<td>gpdma1_ch0_tc</td>
</tr>
<tr>
<td>22</td>
<td>gpdma1_ch1_tc</td>
</tr>
<tr>
<td>23</td>
<td>gpdma1_ch2_tc</td>
</tr>
<tr>
<td>24</td>
<td>gpdma1_ch3_tc</td>
</tr>
<tr>
<td>25</td>
<td>gpdma1_ch4_tc</td>
</tr>
<tr>
<td>26</td>
<td>gpdma1_ch5_tc</td>
</tr>
<tr>
<td>27</td>
<td>gpdma1_ch6_tc</td>
</tr>
<tr>
<td>28</td>
<td>gpdma1_ch7_tc</td>
</tr>
<tr>
<td>29</td>
<td>gpdma1_ch8_tc</td>
</tr>
<tr>
<td>30</td>
<td>gpdma1_ch9_tc</td>
</tr>
<tr>
<td>31</td>
<td>gpdma1_ch10_tc</td>
</tr>
<tr>
<td>HPDMA_CxTR2.TRIGSEL[5:0]</td>
<td>Selected HPDMA trigger</td>
</tr>
<tr>
<td>--------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>32</td>
<td>gpdma1_ch11_tc</td>
</tr>
<tr>
<td>33</td>
<td>gpdma1_ch12_tc</td>
</tr>
<tr>
<td>34</td>
<td>gpdma1_ch13_tc</td>
</tr>
<tr>
<td>35</td>
<td>gpdma1_ch14_tc</td>
</tr>
<tr>
<td>36</td>
<td>gpdma1_ch15_tc</td>
</tr>
<tr>
<td>37</td>
<td>hpdma1_ch0_tc</td>
</tr>
<tr>
<td>38</td>
<td>hpdma1_ch1_tc</td>
</tr>
<tr>
<td>39</td>
<td>hpdma1_ch2_tc</td>
</tr>
<tr>
<td>40</td>
<td>hpdma1_ch3_tc</td>
</tr>
<tr>
<td>41</td>
<td>hpdma1_ch4_tc</td>
</tr>
<tr>
<td>42</td>
<td>hpdma1_ch5_tc</td>
</tr>
<tr>
<td>43</td>
<td>hpdma1_ch6_tc</td>
</tr>
<tr>
<td>44</td>
<td>hpdma1_ch7_tc</td>
</tr>
<tr>
<td>45</td>
<td>hpdma1_ch8_tc</td>
</tr>
<tr>
<td>46</td>
<td>hpdma1_ch9_tc</td>
</tr>
<tr>
<td>47</td>
<td>hpdma1_ch10_tc</td>
</tr>
<tr>
<td>48</td>
<td>hpdma1_ch11_tc</td>
</tr>
<tr>
<td>49</td>
<td>hpdma1_ch12_tc</td>
</tr>
<tr>
<td>50</td>
<td>hpdma1_ch13_tc</td>
</tr>
<tr>
<td>51</td>
<td>hpdma1_ch14_tc</td>
</tr>
<tr>
<td>52</td>
<td>hpdma1_ch15_tc</td>
</tr>
</tbody>
</table>
13.4 **HPDMA functional description**

13.4.1 **HPDMA block diagram**

Figure 108. HPDMA block diagram

13.4.2 **HPDMA channel state and direct programming without any linked-list**

After an HPDMA reset, an HPDMA channel x is in idle state. When the software writes 1 into the HPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (HPDMA_CxXXX), switches to the active/non-idle state, and starts to execute the corresponding requested data transfers.

After enabling/starting an HPDMA channel transfer by writing 1 into HPDMA_CxCR.EN, an HPDMA channel interrupt on a complete transfer notifies the software that the HPDMA channel is back in idle state (EN is then deasserted by hardware), and that the channel is ready to be reconfigured then enabled again.
13.4.3 HPDMA channel suspend and resume

The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or bitfield that is hardware modifiable.
Note: An ongoing HPDMA transfer can be a data transfer (a source/destination burst transfer,) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to HPDMACxCR.SUSP.

**Figure 110. HPDMA channel suspend and resume sequence**

![Diagram showing channel states and suspend/resume sequence]

Note: A suspend and resume sequence does not impact the HPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

### 13.4.4 HPDMA channel abort and restart

Alternatively, like for aborting a continuous HPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master port.
3. The software resets the channel by writing 1 to HPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the HPDMA_CxCR.EN bit, and the reset of the HPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram hardware-modified HPDMA_CxBR1, HPDMA_CxSAR, and HPDMA_CxDAR.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the HPDMA_CxCR.EN bit.

**Figure 111. HPDMA channel abort and restart sequence**

13.4.5 HPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

For a channel x, the base address in memory of the data structure of a next LLI_{n+1} is the sum of the following:
- the link base address of the channel x (in HPDMA_CxLBAR)
- the link address offset LA[15:2] bitfield in HPDMA_CxLLR, that is the updated result from the data structure of the previous LLIn of the channel x

The data structure for each LLI can be specific.

A linked-list data structure is addressed following the value of UT1, UT2, UB1, USA, UDA, and ULL bits, plus UB2 and UT3 when present, in HPDMA_CxLLR.

In linked-list mode, each HPDMA linked-list register (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR or HPDMA_CxLLR,
plus HPDMA_CxTR3 or HPDMA_CxBR2 when present) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of HPDMA_CxLLR that was conditionally updated from the linked-list data structure of the previous LLI.

**Static linked-list data structure**

For example, when the update bits (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present) in HPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x, x = 0 to 11, six contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR (see **Figure 112**) and including the first linked-list register file (LLI0), and the next LLI0s (LLI1, LLI2,...) in the memory
- channel x, x = 12 to 15, eight contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 (see **Figure 113**), including the first linked-list register file (LLI0), and the next LLI0s (LLI1, LLI2,...) in the memory

**Figure 112. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x**
Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of HPDMA_CxLLR points to the updated value of HPDMA_CxTR2, instead of HPDMA_CxTR1 which is not to be modified (see Figure 114).

Example: if UT1 = UB1 = USA = 0, and if UT3 = UB2 = 0 when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for HPDMA_CxTR1, nor HPDMA_CxBR1, nor HPDMA_CxSAR, nor HPDMA_CxTR3, nor HPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for HPDMA_CxTR2, HPDMA_CxDAR, and HPDMA_CxLLR, as shown in Figure 115.
The user must program HPDMA_CxLLR for each LLI_n to be 32-bit aligned, and not to exceed the 64-Kbyte addressable space pointed by HPDMA_CxLBAR.

### 13.4.6 Linked-list item transfer execution

An LLI_n transfer is the sequence of:

1. a data transfer: the HPDMA executes the data transfer as described by the HPDMA internal register file (this data transfer can be void/null for LLI_0).
2. a conditional link transfer: the HPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the HPDMA_CxLLR value of the LLI_n.

**Note:** The initial data transfer, as defined by the internal register file (LLI_0), can be null (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxTR2.PFREQ = 0), provided that UB1 is set in HPDMA_CxLLR (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended HPDMA use, an HPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, HPDMA_CxCR.LSM = 0), or can be programmed for a single execution of a LLI (link step mode, HPDMA_CxCR.LSM = 1), as described in the next sections.
13.4.7 HPDMA channel state and linked-list programming in run-to-completion mode

When HPDMA_CxCR.LSM = 0 (in full-list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), an HPDMA channel x is initially programmed, started by writing 1 to HPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI0, internal linked-list register file: HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present
  - the last LLI_{N-1} described by the linked-list data structure in memory, as defined by HPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when HPDMA_CxLLR[31:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if BRC[10:0] is present and HPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last transfer

HPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion, and means the following:

- The 16 low significant bits HPDMA_CxLLR.LA[15:0] of the next link address are null.
- All bits HPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA,UDA, and ULL, plus UB2 and UT3 when present).

The channel may never be completed when HPDMA_CxLLR.LSM = 0:

- If the last LLI_{N-1} is recursive, pointing to itself as a next LLI, in one of the following:
  - HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated by the same value.
  - HPDMA_CxLLR.ULL = 0
- If LLI_{N} points to a previous LLI

In the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see Section 13.4.14 for more details).

In the typical run-to-completion mode, the allocation of an HPDMA channel, including its fine programming, is done once during the HPDMA initialization. In order to have a reserved data communication link and HPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 116 depicts the HPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 116 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel...
In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 116, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 13.3.5).
Figure 116. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0)

Channel state = Idle
- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active
- Valid user setting?
  - Yes: Execute once the data transfer from the register file
  - No: Set TCF = 1
- BNDT ≠ 0?
  - Yes: Execute once the data transfer from the register file
  - No: Set USEF = 1
- No transfer error?
  - Yes: Loading next LLI into the register file
  - No: Setting DTEF = 1
- LLR ≠ 0?
  - Yes: Setting ULEF = 1
  - No: Valid user setting?
    - Yes: Setting USEF = 1
    - No: Setting TCF = 1
- No transfer error?
  - Yes: Setting USEF = 1
  - No: Setting USEF = 1
- Valid user setting?
  - Yes: Setting USEF = 1
  - No: Setting USEF = 1

End
Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following bitfields of HPDMA_CxTR2 in the data structure of the LLI_{n-1}:

- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[5:0] (see Section 13.3.6 for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory, and to generate a transfer complete event hpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 117 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

**Caution:** This use case is restricted to an application with an LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.
13.4.8 HPDMA channel state and linked-list programming in link step mode

When HPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

An HPDMA channel transfer can be programmed at an LLI level, started by writing 1 into HPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI\(_n\) transfer is described with the following:
  - HPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - HPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]), for channel \(x = 12\) to \(15\), the number of blocks at a 2D/repeated block level (BRC[10:0]+1), and the incrementing/decrementing mode for address offsets.
- HPDMA_CxTR2 defines the input control (request, trigger), and the output control (transfer-complete event) of the transfer.
- HPDMA_CxSAR and HPDMA_CxDAR define the source/destination transfer start address.
- HPDMA_CxTR3 (for channel x = 12 to 15) defines the source/destination additional address offset between burst transfers.
- HPDMA_CxBR2 (for channel x = 12 to 15) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
- HPDMA_CxLLR defines the data structure, and the address offset of the next LLI_{n+1} in the memory.

- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when HPDMA_CxBR1.BRC[10:0] = 0 if BRC[10:0] is present, and HPDMA_CxBR1.BNDT[15:0] = 0)
  - after the (conditional) update of the HPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If an LLI is recursive (pointing to itself as a next LLI, either HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated by the same value, or HPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer can be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see Section 13.4.14 for more details).

The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of HPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 118 depicts the HPDMA channel execution mode, and its programming in link step mode.

Note: Figure 118 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 118, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 13.3.5).
Figure 118. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1)

Channel state = Idle
- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active
- Valid user setting?
  - Yes
    - BNDT ≠ 0?
      - Yes
        - Execute once the data transfer from the register file
      - No
        - No transfer error?
          - Yes
            - LLR ≠ 0?
              - Yes
                - Loading next LLI into the register file
              - No
                - No transfer error?
                  - Yes
                    - Valid user setting?
                      - Yes
                        - Setting TCF = 1 Disabling DMA channel
                      - No
                        - Setting USEF = 1 Disabling DMA channel
                    - No
                      - Setting USEF = 1 Disabling DMA channel
                  - No
                    - Setting ULEF = 1 Disabling DMA channel
          - No
            - Setting USEF = 1 Disabling DMA channel
        - No
          - Setting DTEF = 1 Disabling DMA channel
  - No
    - End
Run-time adding a $\text{LLI}_{n+1}$ in link step mode

During run-time, the software can defer the elaboration of the $\text{LLI}_{n+1}$ (and next LLIs), until/after the HPDMA executed the transfer from the $\text{LLI}_{n-1}$ and loaded the $\text{LLI}_n$ from the memory, as shown in the figure below.

**Figure 119. Building $\text{LLI}_{n+1}$: HPDMA dynamic linked-lists in link step mode**

Run-time replacing an $\text{LLI}_n$ with a new $\text{LLI}_n'$ in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new $\text{LLI}_n'$, after the HPDMA executed the transfer from the $\text{LLI}_{n-1}$, and loaded a formerly elaborated $\text{LLI}_n$ from the memory by overwriting directly the linked-list register file with the new $\text{LLI}_n'$, as shown in **Figure 120**.
LSM = 1 with 1-stage linked-list programming:
Overwriting the (pre)loaded LLIₙ linked-list register file with
a new LLIₙ', directly in linked-list register file.
DMA executes LLIₙ₋₁ and load LLIₙ, then CPU builds and overwrites LLIₙ'.

Figure 120. Replace with a new LLIₙ' in register file in link step mode
Run-time replacing an LLI_n with a new LLI_n' in link step mode (in the memory)

The software can build and insert a new LLI_n' and LLI_{n+1}' in the memory, after HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (HPDMA_CxBR1.BNDT[15:0] to be null, and HPDMA_CxLLR to point to new LLI_n) as shown in Figure 121.

Figure 121. Replace with a new LLI_n' and LLI_{n+1}' in memory in link step mode (option 1)
Run-time replacing a $\text{LLI}_n$ with a new $\text{LLI}_{n'}$ in link step mode

Other software implementations exist. Meanwhile the HPDMA executes the transfer from the $\text{LLI}_{n-1}$ and loads a formerly elaborated $\text{LLI}_n$ from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $\text{LLI}_n$ and a new $\text{LLI}_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for $\text{LLI}_{n-1}$ is then restricted to overwrite HPDMA_CxBR1.BNDT[15:0] to be null and HPDMA_CxLLR to point to new $\text{LLI}_{n'}$, as shown in the figure below.

Figure 122. Replace with a new $\text{LLI}_{n'}$ and $\text{LLI}_{n+1'}$ in memory in link step mode (option 2)

---

LSM = 1 with 1-stage linked-list programming:
Overwriting the (pre)loaded $\text{LLI}_n$ linked-list register file by building new $\text{LLI}_{n'}$ and $\text{LLI}_{n+1'}$ in memory while disabling (temporary) channel interrupt at NVIC level, and overwriting DMA_CxBR1.BNDT = 0 and DMA_CxLLR to point to new $\text{LLI}_{n'}$

DMA executes $\text{LLI}_{n-1}$ and loading $\text{LLI}_n$ while CPU builds ($\text{LLI}_{n'}$ and $\text{LLI}_{n+1'}$), then CPU overwrites (BR1 and LLR)
13.4.9 HPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (HPDMA_CxCR.EN = 0), and update the execution mode (HPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the HPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the HPDMA channel
- abort and reconfigure the channel with an LSM update (see Section 13.4.4.)

In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

*Figure 123* shows the overall and unified HPDMA linked-list programming, whatever is the execution mode.

*Note:* *Figure 123* is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

*In Figure 123, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 13.3.5).*
Figure 123. HPDMA channel execution and linked-list programming

Channel state = Idle

- Initialize DMA channel
- Enable DMA channel
- Reconfigure DMA channel

Channel state = Active

- Valid user setting?
- BNDT ≠ 0?
- Executing once the data transfer from the register file
- No transfer error?
- LLR ≠ 0?
- Loading next LLI into the register file
- No transfer error?
- Valid user setting?
- LSM = 1?
- Setting TCF = 1 Disabling DMA channel
- Setting USEF = 1 Disabling DMA channel
- Setting DTEF = 1 Disabling DMA channel
- Setting ULEF = 1 Disabling DMA channel

End
13.4.10 HPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel $x$ is implemented with a dedicated FIFO whose size is defined by $\text{dmafifo}_x$ (see Section 13.3.1 for more details).

HPDMA burst

A programmed transfer at the lowest level is an HPDMA burst.

An HPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by $\text{SBL}_1[5:0]$ (respectively $\text{DBL}_1[5:0]$), and with a data width defined by $\text{SDW}_2[1:0]$ (respectively $\text{DDW}_2[1:0]$) in HPDMA_CxTR1.

The addressing mode after each data (named beat) of an HPDMA burst is defined by SINC and DINC in HPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of an HPDMA source/destination burst (defined by HPDMA_CxSAR and HPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of an HPDMA burst.

<table>
<thead>
<tr>
<th>SAP/DAP (allocated port)</th>
<th>SDW_LOG2[1:0]</th>
<th>Data width (bytes)</th>
<th>SINC/DINC</th>
<th>SBL_1[5:0]</th>
<th>DBL_1[5:0]</th>
<th>Burst length (data/beats)</th>
<th>Next data/beat address</th>
<th>Next burst address</th>
<th>Burst address alignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: AXI 1: AHB</td>
<td>00</td>
<td>1</td>
<td>0 (fixed)</td>
<td></td>
<td></td>
<td>+0</td>
<td>+0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>+1</td>
<td>(n+1)</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>+2</td>
<td>(n+1)</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>0: AXI</td>
<td>11</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>+4</td>
<td>(n+1)</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>0: AXI 1: AHB</td>
<td>00</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>+1</td>
<td>(n+1)</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>01</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td>+2</td>
<td>(n+1)</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td>+4</td>
<td>(n+1)</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>+8</td>
<td>(n+1)</td>
<td></td>
<td>8</td>
</tr>
</tbody>
</table>

1. When $\text{S/DBL}_1[5:0] = 0$, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by HPDMA_CxSAR or HPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.
HPDMA burst with 2D addressing (channel x = 12 to 15)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by HPDMA_CxTR2.SAO[12:0]/DAO[12:0] and HPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by HPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and HPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by Figure 124 with a 2D destination buffer.
Programmable address jumps 1) after burst and 2) after block.

Example:
burst: \( I \) * words (DBL_1=I-1; DDW_LOG2='b10)
block: \( J \) * bursts (BNDT=J\(^4\))
LLI: \( K \) * blocks (BRC=K-1)
HPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by HPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by HPDMA_CxTR1.DAP

HPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

\[
\text{when FIFO level} \leq 2^{\text{dma_fifo_size}[x]} - (\text{SBL}_1[5:0]+1) \times 2^{\text{SDW_LOG2}[1:0]}
\]

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- \(2^{\text{dma_fifo_size}[x]}\) is the channel x FIFO size, in bytes (see Section 13.3.1 for the implementation details and dma_fifo_size[x] value).
- \((\text{SBL}_1[5:0]+1) \times 2^{\text{SDW_LOG2}[1:0]}\) is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (HPDMA_CxCR.PRI0[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

HPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

\[
\text{when FIFO level} \geq (\text{DBL}_1[5:0]+1) \times 2^{\text{DDW_LOG2}[1:0]}
\]

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- \((\text{DBL}_1[5:0]+1) \times 2^{\text{DDW_LOG2}[1:0]}\) is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

HPDMA burst versus source block size, 1- or 4-Kbyte address boundary and FIFO size

The programmed source/destination HPDMA burst is implemented with an AHB/AXI burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination HPDMA burst is implemented with a series of singles or bursts of a lower size, each transfer being of a size that is lower or equal than half of the FIFO size, without any user constraint.
- if the source block size (HPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (HPDMA_CxTR1.SDW_LOG2[1:0]), the HPDMA modifies and shortens bursts into
singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1- or 4-Byte address boundary on, respectively, an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.

- If the source/destination burst length exceeds 16 on an AHB transfer, or if the source/destination burst on an AXI transfer is both with fixed addressing and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.

In any case, the HPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in HPDMA_CxSR) is compared to and updated with the effective transfer size, and the HPDMA re-arbitrates between each AHB/AXI single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the HPDMA read transfers related to the update of the linked-list parameters from the memory to the internal HPDMA registers, are scheduled over the link allocated port, as programmed by HPDMA_CxCR.LAP.

HPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension, and left/right alignment

The data handling is controlled by HPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word, word, or double-word, as per SDW_LOG2[1:0] and DDW_LOG2[1:0] (see Table 108).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if the source data width is a word or a double-word (for AXI source bus, SAP = 0), the two bytes of the unaligned half-word at the middle of each source data word are exchanged.

2. Data-width conversion by packing, unpacking, padding, or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.
   - If DWX = 1 and if the destination data width is a double-word and the destination bus is AXI (DAP = 0), the two aligned words are exchanged within aligned (post PAM[1:0]) double-words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding, and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(1)</th>
<th>SBX</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM<a href="2">1:0</a></th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Byte</td>
<td>xx</td>
<td>x</td>
<td>00 (RA, 0P)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>00 (RA, SE)(3)(4)</td>
<td>1</td>
<td>01 (RA, SE)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>1x (PACK)</td>
<td>x</td>
<td>00 (RA, 0P)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Byte</td>
<td>B7, B6, B5, B4, B3, B2, B1, B0</td>
<td>x</td>
<td>00 (RA, 0P)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>00 (RA, SE)(3)(4)</td>
<td>1</td>
<td>01 (RA, SE)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>1x (PACK)</td>
<td>x</td>
<td>00 (RA, 0P)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Byte</td>
<td>B7, B6, B5, B4, B3, B2, B1, B0</td>
<td>x</td>
<td>00 (RA, 0P)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>00 (RA, SE)(3)(4)</td>
<td>1</td>
<td>01 (RA, SE)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>1x (PACK)</td>
<td>x</td>
<td>00 (RA, 0P)(3)(4)</td>
<td>0</td>
<td>00 B7,B6,B5,B4,B3, B2, B1, B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Source data stream
(2) PAM[1:0] = 00: 0000h; 01: 0001h; 10: 1111h; 11: 1110h
(3) RA: Read address; 0P: 0000h; SE: 0001h
(4) B0, B1, B2, B3, B4, B5, B6, B7 = data bits

The table above lists the possible data handling from the source to the destination.
### Table 108. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Byte</td>
<td>(B_7, B_6, B_5, B_4, B_3, B_2, B_1, B_0)</td>
<td>x</td>
<td>11(^{(5)})</td>
<td>Double-word</td>
<td>00 (RA, 0P)(^{(3)(4)})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00000000B_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>000000B_000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00000B_0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>000B_000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0B_00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_000000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (RA, SE)(^{(3)(4)})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>SSSSSSSB_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SSSSSSB_0S</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>SSSSB_0SSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>SSSB_0SSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>SB_0SSSSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_0SSSSSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (PACK)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_7B_6B_5B_4B_3B_2B_1B_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_6B_5B_4B_3B_2B_1B_0B_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B_5B_4B_3B_2B_1B_0B_0B_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_4B_3B_2B_1B_0B_0B_1B_2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_3B_2B_1B_0B_0B_0B_2B_3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_2B_1B_0B_0B_0B_1B_2B_4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B_1B_0B_0B_0B_1B_2B_4B_6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B_0B_1B_2B_3B_4B_6B_7</td>
</tr>
<tr>
<td>SDW_LOG2 [1:0]</td>
<td>Source data</td>
<td>Source data stream (1)</td>
<td>SBX</td>
<td>DDW_LOG2 [1:0]</td>
<td>Destination data</td>
<td>PAM[1:0]</td>
<td>DBX</td>
<td>DHX</td>
<td>DWX</td>
<td>Destination data stream (1)</td>
</tr>
<tr>
<td>----------------</td>
<td>-------------</td>
<td>------------------------</td>
<td>-----</td>
<td>----------------</td>
<td>-----------------</td>
<td>----------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>----------------------------</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td></td>
<td></td>
<td>x</td>
<td>00 (RA, LT)</td>
<td>x</td>
<td></td>
<td></td>
<td>B6B4B2B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00</td>
<td>Byte</td>
<td>01 (LA, RT)</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>B7B6B3B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B7B6B5B4B3B2B1B0, B0</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>xx</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B7B6B5B4B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B6B7B4B5B2B3B0B1</td>
</tr>
<tr>
<td>10</td>
<td>Word</td>
<td>x</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>00B3B2B0B0B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B3B200B1B00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>00B2B3B00B3B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B2B300B1B00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>SSBB3B2B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B2B3BB1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B3BB2BB0B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>SSBB2B3BB0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>B7B6B5B4B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B7B6B5B4B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>B5B4B7B6B1B0B3B2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>B4B5B6B7B5B3B2B3</td>
</tr>
</tbody>
</table>
### Table 108. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW _ LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW _ LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Half-word</td>
<td>(B_7B_6B_5B_4, B_3B_2B_1B_0)</td>
<td>x</td>
<td>11(^{(5)})</td>
<td>Double-word</td>
<td>00 (RA, 0P)(^{(3),(4)})</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>000000B_1B_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>000000B_0B_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0000B_1B_0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0000B_000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (RA, SE)(^{(3),(4)})</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>B_1B_0000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B_0B_1000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>SSSSSSSB_1B_0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>SSSSSSSB_0B_1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>SSSSSSB_0SS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>SSSSSB_0SSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>SBB_1B_0SSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>SSSB_1B_0SSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>SSSSSSSB_0SSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>SSSSSSBSB_0SSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>B_1B_0SSSSSS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B_0B_1SSSSSS</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Depending on the source data type.

\(^{(2)}\) Depending on the destination data type.

\(^{(3)}\) 00 is the default value.

\(^{(4)}\) 01 is the default value.

\(^{(5)}\) \(x\) indicates that any data can be used.

1x (PACK) indicates that the data is packed.

SDW and DDW are logic 2 bits that represent the number of data words.

PAM[1:0] represents the programmed access mode.

DBX, DHX, and DWX are logic 3 bits that represent the data type.

Destination data stream represents the destination data stream that can be selected.

Source data stream represents the source data stream that can be selected.
### Table 108. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2[1:0]</th>
<th>Source data</th>
<th>Source data stream(1)</th>
<th>SBX</th>
<th>DDW_LOG2[1:0]</th>
<th>Destination data</th>
<th>PAM<a href="2">1:0</a></th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Word</td>
<td>B7B6B5B4, B3B2B1B0</td>
<td>0</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)(3)</td>
<td>-</td>
<td></td>
<td></td>
<td>B12,B6,B4,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)(3)</td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td>B15,B11,B7,B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td>B7,B6,B5,B4,B3,B2,B1, B0</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td></td>
<td>0</td>
<td>00 (RA, LT)(3)</td>
<td></td>
<td>0</td>
<td></td>
<td>x</td>
<td></td>
<td>B5B4,B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)(3)</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B4B5,B0B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>B6B7,B2B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>B7B6,B5B4,B3B2,B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B6B7,B4B5,B2B3,B0B1</td>
</tr>
<tr>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>B7B6B5B4,B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B6B7B5B4,B2B3B0B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>B2B4B7B6,B1B0B2B3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>B4B5B6B7,B5B1B2B3</td>
</tr>
</tbody>
</table>

(1) B: Bit, S: Source, D: Destination
<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B(_7)B(_6)B(_5)B(_4), B(_3)B(_2)B(_1)B(_0)</td>
<td>Double-word</td>
<td>0</td>
<td>01(^{(5)})</td>
<td>1x (PACK)</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>B(_7)B(_6)B(_5)B(_4)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 1</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>B(_5)B(_4)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>B(_3)B(_2)B(_1)B(_0)B(_5)B(_4)B(_3)B(_2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 1</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>B(_2)B(_1)B(_0)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0</td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>B(_1)B(_0)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>B(_0)B(_1)B(_2)B(_3)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>B(_7)B(_6)B(_5)B(_4)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>B(_5)B(_4)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>B(_3)B(_2)B(_1)B(_0)B(_5)B(_4)B(_3)B(_2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 1</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>B(_2)B(_1)B(_0)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>1 1</td>
<td>B(_1)B(_0)B(_3)B(_2)B(_1)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>1 0</td>
<td>B(_0)B(_1)B(_2)B(_3)B(_1)B(_0)</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Source data stream: SSB = Source data stream bit, SW = Source word bit.

\(^{(2)}\) PAM[1:0]: PAM = Programmed Address Mode, M[1:0] = Memory mode.[3]  \(^{(3)}\) Source data stream: SSB = Source data stream bit, SW = Source word bit.

\(^{(4)}\) DBX, DHX, DWX: DB = Double-bit, DH = Double-Hex, DW = Double-Wide.

\(^{(5)}\) DBX, DHX, DWX: DB = Double-bit, DH = Double-Hex, DW = Double-Wide.
<table>
<thead>
<tr>
<th>SDW_{LOG2 [1:0]}</th>
<th>Source data</th>
<th>Source data stream^{(1)}</th>
<th>SBX</th>
<th>DDW_{LOG2 [1:0]}</th>
<th>Destination data</th>
<th>PAM[1:0]^{(2)}</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream^{(1)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Word</td>
<td>B_{7}B_{6}B_{5}B_{4}, B_{3}B_{2}B_{1}B_{0}</td>
<td>1</td>
<td>00</td>
<td>Byte</td>
<td>00 (RA, LT)^{(3)}</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>B_{12}B_{11}B_{4}B_{0}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td></td>
<td>01 (LA, RT)^{(3)}</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>B_{15}B_{11}B_{7}B_{3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
<td>B_{7}B_{6}B_{5}B_{4},B_{3}B_{1},B_{2},B_{0}</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td></td>
<td></td>
<td>00</td>
<td></td>
<td>00 (RA, LT)^{(3)}</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>B_{6}B_{4}B_{2}B_{0}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01</td>
<td></td>
<td>01 (LA, RT)^{(3)}</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B_{7}B_{5}B_{3}B_{1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>B_{5}B_{7}B_{1}B_{3}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>B_{7}B_{5}B_{3}B_{1},B_{2}B_{0}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td>B_{5}B_{7}B_{4}B_{6}B_{1}B_{3},B_{0}B_{2}</td>
</tr>
<tr>
<td>10</td>
<td>Word</td>
<td>xx</td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, LT)^{(3)}</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B_{6}B_{4}B_{2}B_{0}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)^{(3)}</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B_{7}B_{5}B_{3}B_{1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td></td>
<td>B_{5}B_{7}B_{4}B_{6}B_{1}B_{3},B_{0}B_{2}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td>B_{6}B_{4}B_{7}B_{5},B_{2}B_{0},B_{1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td>B_{4}B_{6}B_{5}B_{7},B_{2}B_{1}B_{3}</td>
</tr>
</tbody>
</table>

^{(1)} Source data streams and destination data streams are defined by the specific configurations of SDW_{LOG2 [1:0]} and DDW_{LOG2 [1:0]}.

^{(2)} PAM[1:0] represents the programmed data handling mode, where 00 indicates Byte, 01 indicates Half-word, and 10 indicates Word.
<table>
<thead>
<tr>
<th>SDW_ LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW_ LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Word</td>
<td>(B_7B_6B_5B_4, B_3B_2B_1B_0)</td>
<td>1</td>
<td>11(^{(5)})</td>
<td>00 (RA, 0P)(^{(3)(4)}) 0</td>
<td>0</td>
<td>0</td>
<td>0000B_1B_2B_0B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (RA, SE)(^{(3)(4)}) 1</td>
<td>1</td>
<td>0</td>
<td>0000B_1B_2B_0B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>0000B_2B_3B_1B_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0000B_2B_3B_1B_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>B_3B_1B_2B_00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_3B_1B_2B_00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>B_2B_0B_3B_10000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_2B_0B_3B_10000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>SSSSB_3B_1B_2B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>SSSSB_3B_1B_2B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>SSSSB_2B_0B_3B_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>SSSSB_2B_0B_3B_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>B_3B_1B_2B_0SSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_3B_1B_2B_0SSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>B_2B_0B_3B_1SSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_2B_0B_3B_1SSS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>B_7B_6B_5B_4B_3B_2B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_7B_6B_5B_4B_3B_2B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>B_6B_5B_4B_3B_2B_1B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_6B_5B_4B_3B_2B_1B_0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>B_3B_1B_2B_0B_2B_4B_7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_3B_1B_2B_0B_2B_4B_7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>B_2B_0B_3B_1B_4B_6B_5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>B_2B_0B_3B_1B_4B_6B_5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\(^{(1)}\) Source data stream: \(S_x\) = \(S_{x-1}\) \& PAM \& SBX, i.e., \(S_x = (S_{x-1} \& PAM \& SBX) \& D_x\)

\(^{(2)}\) PAM = \(PAM[1:0]\) = \(D_x = \{D_x[1:0]\}

\(^{(3)}\) SDW_ LOG2 = \(SDW_ LOG2[1:0]\) = \(PAM[0:0]\)

\(^{(4)}\) DDW_ LOG2 = \(DDW_ LOG2[1:0]\) = \(PAM[1:0]\)

\(^{(5)}\) SBX = \(SBX[1:0]\) = \(PAM[1:0]\)

\(^{(6)}\) PAM = \(PAM[1:0]\) = \(SBX[0:0]\)

\(^{(7)}\) DBX = \(DBX[1:0]\) = \(PAM[1:0]\)

\(^{(8)}\) DHX = \(DHX[1:0]\) = \(PAM[1:0]\)

\(^{(9)}\) DWX = \(DWX[1:0]\) = \(PAM[1:0]\)
### Table 108. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>Byte</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, LT)(^{(3)})</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)(^{(3)})</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK) x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, LT)(^{(3)})</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)(^{(3)})</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK) 0</td>
<td>1</td>
<td></td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td>11(^{(6)})</td>
<td>Double-word</td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, LT)(^{(3)})</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)(^{(3)})</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK) 0</td>
<td>1</td>
<td></td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td>10</td>
<td>Word</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00 (RA, LT)(^{(3)})</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01 (LA, RT)(^{(3)})</td>
<td>0</td>
<td>1</td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1x (UNPACK) 0</td>
<td>1</td>
<td></td>
<td></td>
<td>B(_7)B(_8)B(_9)B(_0)</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Source data stream refers to the data flow direction.

\(^{(2)}\) PAM[1:0] determines the data handling mode.

\(^{(3)}\) RA, LT, LA, RT refer to the registers involved in the data transmission.
### Table 108. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>11(^{(6)})</td>
<td>Double-word</td>
<td>01 11(^{(5)})</td>
<td></td>
<td>0</td>
<td>Double-word</td>
<td>xx</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B7B6B5B4B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td>B7B6B5B4B3B2</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>B5B4B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td>B1B0</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>B4B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B3B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>B2B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B1B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>B0B1</td>
</tr>
<tr>
<td>00</td>
<td>Byte</td>
<td>01 (RA, LT)(^{(3)})</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>x</td>
<td>B24,B16,B8,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01 (LA, RT)(^{(3)})</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>B31,B23,B15,B7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>B7,B5,B3,B1,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>B5,B3,B1,B0</td>
</tr>
<tr>
<td>01</td>
<td>Half-word</td>
<td>00 (RA, LT)(^{(3)})</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>B10,B8,B6,B0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01 (LA, RT)(^{(3)})</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>B15,B13,B7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>x</td>
<td>B7,B5,B3,B1,B2,B0</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Source data stream: 00, 01, or 1x

\(^{(2)}\) PAM[1:0]: 00, 01, or 1x

\(^{(3)}\) RA, LT, LA, or RT

\(^{(4)}\) DBX, DHX, or DWX

\(^{(5)}\) Double-word

\(^{(6)}\) Double-word

\(^{(7)}\) Source data: B7B6B5B4B3B2B1B0

\(^{(8)}\) Destination data: B7B6B5B4B3B2B1B0
### Table 108. Programmed data handling (continued)

<table>
<thead>
<tr>
<th>SDW_LOG2 [1:0]</th>
<th>Source data</th>
<th>Source data stream(^{(1)})</th>
<th>SBX</th>
<th>DDW_LOG2 [1:0]</th>
<th>Destination data</th>
<th>PAM[1:0](^{(2)})</th>
<th>DBX</th>
<th>DHX</th>
<th>DWX</th>
<th>Destination data stream(^{(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>11(^{(4)}) Double-word</td>
<td>B7B6B5B4B3B2B1B0</td>
<td>10 Word</td>
<td>00 (RA, LT)(^{(3)})</td>
<td>0</td>
<td>0</td>
<td>B11B9B10B8B3B1B2B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B2B11B9B10B8B3B2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>B12B2B1B3B5B6B7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B3B12B1B3B5B6B7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1x (UNPACK)</td>
<td>0</td>
<td>0</td>
<td>B7B6B5B4B3B1B2B0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B2B7B6B5B4B3B2B1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>B3B2B7B6B5B4B3B2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B4B3B2B7B6B5B4B3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11(^{(5)}) Double-word</td>
<td>xx</td>
<td>10 Word</td>
<td>01 (LA, RT)(^{(3)})</td>
<td>0</td>
<td>0</td>
<td>B13B14B12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B7B4B5B6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>B15B12B13B15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B8B4B5B6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>B14B12B13B15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B6B8B4B5B6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>B3B14B12B13B15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B2B3B14B12B13B15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>B4B3B14B12B13B15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>B0B4B3B14B12B13B15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Data stream is timely ordered starting from the byte with the lowest index (B0).
2. RA = right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.
3. RA = right aligned, LA = left aligned, RT = right truncated, LT = left truncated.
4. 0P = zero-bit padding up to the destination data width, SE = sign bit extended up to the destination data width.
5. If DDW\_LOG2[1:0] = 11 and if DAP = 0 (destination allocated port is AXI). Else if DDW\_LOG2[1:0] = 11 and DAP = 1 (AHB), a user setting error (USEF) is reported.
6. If SDW\_LOG2[1:0] = 11 and if SAP = 0 (source allocated source port is AXI). Else if SDW\_LOG2[1:0] = 11 and SAP = 1 (AHB), a user setting error (USEF) is reported.
13.4.11 HPDMA transfer request and arbitration

HPDMA transfer request

As defined by HPDMA_CxTR2, a programmed HPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the HPDMA hardware peripheral request is driven by REQSEL[4:0] (see Section 13.3.4). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in HPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control register update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign the same input hardware peripheral HPDMA request via HPDMA_CxTR.REQSEL[4:0] to two different channels, if at a given time this request is asserted by the peripheral, and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

HPDMA transfer request for arbitration

A ready FIFO-based HPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (HPDMA_CxTR1.SAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIOR[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

A ready FIFO-based HPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (HPDMA_CxTR1.DAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIOR[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

An arbitrated HPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CsAR, HPDMA_CsAR, HPDMA_CxTR3, HPDMA_CxBR2 or HPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (HPDMA_CxCR.LAP).

Whenever the requested data transfer is programmed with a software request for a memory-to-memory transfer (HPDMA_CxTR2.SWREQ = 1), or with a hardware request (HPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (HPDMA_CxTR2.SWREQ = 0 and HPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request
to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

HPDMA arbitration

The HPDMA arbitration is directed from the 4-grade assigned channel priority (HPDMA_CxCR.PRIOR[1:0]). The arbitration policy, as illustrated in Figure 125, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels.
  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.

- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2.
  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

**Figure 125. HPDMA arbitration policy**

HPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- equal maximum bandwidth between requests with same priority
- reserved bandwidth (noted as BQ3) to the time-sensitive requests (with priority 3)
- residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).
The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive HPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0\(B_{\text{req0}}\) is equal to the allocated bandwidth to req1\(B_{\text{req1}}\).
    \[ B_{\text{req0}} = B_{\text{req1}} = \frac{1}{2} \cdot (1 - B_{Q3}) \]
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0\(B_{P0}\) is 3 times less than the allocated bandwidth to req1\(B_{P1}\).
    \[ B_{\text{req0}} = B_{P0} = \frac{1}{2} \cdot \frac{1}{2} \cdot (1 - B_{Q3}) = \frac{1}{4} \cdot (1 - B_{Q3}) \]
    \[ B_{\text{req1}} = B_{P1} = (\frac{1}{2} + 1) \cdot \frac{1}{2} \cdot (1 - B_{Q3}) = \frac{3}{4} \cdot (1 - B_{Q3}) \]
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0\(B_{P0}\) is 5 times less than the allocated bandwidth to req1\(B_{P2}\).
    \[ B_{\text{req0}} = B_{P0} = \frac{1}{2} \cdot \frac{1}{3} \cdot (1 - B_{Q3}) = \frac{1}{6} \cdot (1 - B_{Q3}) \]
    \[ B_{\text{req1}} = B_{P2} = (\frac{1}{2} + 1 + 1) \cdot \frac{1}{3} \cdot (1 - B_{Q3}) = \frac{5}{6} \cdot (1 - B_{Q3}) \]

The above computed bandwidth calculation is based on a theoretical input request, always active for any HPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one HPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of HPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0\(B_{P0}\) is
    \[ B_{P0} = \frac{1}{3} \cdot \frac{1}{3} \cdot (1 - B_{Q3}) = \frac{1}{9} \cdot (1 - B_{Q3}) \]
  - The allocated bandwidth to the set of requests of priority 1\(B_{P1}\) is
    \[ B_{P1} = \left(\frac{1}{3} + \frac{1}{2}\right) \cdot \frac{1}{3} \cdot (1 - B_{Q3}) = \frac{5}{18} \cdot (1 - B_{Q3}) \]
  - The allocated bandwidth to the set of requests of priority 2\(B_{P2}\) is
    \[ B_{P2} = \left(\frac{1}{3} + \frac{1}{2} + 1\right) \cdot \frac{1}{3} \cdot (1 - B_{Q3}) = \frac{11}{18} \cdot (1 - B_{Q3}) \]
  - The allocated bandwidth to any request n\(B_{n}\) among the N requests of that priority Pi (i = 0 to 2) is
    \[ B_{n} = \frac{1}{N} \cdot B_{Pi} \]
    \[ B_{n, P0} = \frac{1}{N} \cdot \frac{1}{9} \cdot (1 - B_{Q3}) \]
    \[ B_{n, P1} = \frac{1}{N} \cdot \frac{5}{18} \cdot (1 - B_{Q3}) \]
    \[ B_{n, P2} = \frac{1}{N} \cdot \frac{11}{18} \cdot (1 - B_{Q3}) \]

In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.
More generally, assume that the following requests are present:

- I requests \((I \geq 0)\) assigned to priority 0
  
  If \(I > 0\), these requests are noted from \(i = 0\) to \(I-1\).

- J requests \((J \geq 0)\) assigned to priority 1
  
  If \(J > 0\), these requests are noted from \(j = 0\) to \(J-1\).

- K requests \((K > 0)\) assigned to priority 2
  
  These requests are noted from \(k = 0\) to \(K-1\).

- L requests \((L \geq 0)\) assigned to priority 3
  
  If \(L > 0\), these requests are noted from \(l = 0\) to \(L-1\).

As \(B_Q3\) is the reserved bandwidth to time-sensitive requests, the bandwidth for each request \(L\) with priority 3 is:

- \(B_L = B_Q3 / L\) for \(L > 0\) (else: \(B_L = 0\))

The bandwidth for each non-time sensitive queue is:

- \(B_Q0 = 1/3 \ast (1 - B_Q3)\)
- \(B_Q1 = 1/3 \ast (1 - B_Q3)\)
- \(B_Q2 = 1/3 \ast (1 - B_Q3)\)

The bandwidth for the set of requests with priority 0 is:

- \(B_{P0} = I / (I + J + K) \ast B_Q0\)

The bandwidth for each request \(i\) with priority 0 is:

- \(B_i = B_{P0} / I\) for \(L > 0\) (else \(B_i = 0\))

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- \(B_{P1,Q0} = J / (I + J + K) \ast B_Q0\)

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- \(B_{P1,Q1} = J / (J + K) \ast B_Q1\)

The total bandwidth for the set of requests with priority 1 is:

- \(B_P1 = B_{P1,Q0} + B_{P1,Q1}\)

The bandwidth for each request \(j\) with priority 1 is:

- \(B_j = B_P1 / J\) for \(J > 0\) (else \(B_j = 0\))

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- \(B_{P2,Q0} = K / (I + J + K) \ast B_Q0\)

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- \(B_{P2,Q1} = K / (J + K) \ast B_Q1\)

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- \(B_{P2,Q2} = B_Q2\)

The total bandwidth for the set of requests with priority 2 is:

- \(B_P2 = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}\)

The bandwidth for each request \(k\) with priority 2 is:

- \(B_k = B_P2 / K\) (K>0 in the general case)
Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - \( B_i = \frac{1}{I} \times \frac{1}{3} \times \frac{I}{I + J + K} \times (1 - B_{Q3}) \)
  - \( B_j = \frac{1}{J} \times \frac{1}{3} \times \left[\frac{J}{I + J + K} + \frac{J}{J + K}\right] \times (1 - B_{Q3}) \)
  - \( B_k = \frac{1}{K} \times \frac{1}{3} \times \left[\frac{K}{I + J + K} + \frac{K}{J + K} + 1\right] \times (1 - B_{Q3}) \)
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - \( B_i = \frac{1}{I} \times \frac{1}{2} \times \frac{I}{I + J} \times (1 - B_{Q3}) \)
  - \( B_j = \frac{1}{J} \times \frac{1}{2} \times \left[\frac{J}{I + J} + 1\right] \times (1 - B_{Q3}) \)

Consequently, the HPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

### 13.4.12 HPDMA triggered transfer

A programmed HPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by HPDMA_CxTR2.TRIGPOL[1:0] and HPDMA_CxTR2.TRIGSEL[5:0] (see Section 13.3.6 for the trigger selection).

The triggered transfer, as defined by the trigger mode in HPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block (for channel \( x = 12 \) to 15), or each programmed burst read/write. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in HPDMA channel x transfer register 2 (HPDMA_CxTR2) for more details).

#### Trigger hit memorization and trigger overrun flag generation

The HPDMA monitoring of a trigger for a channel \( x \) is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[5:0] are not modified, and the channel is enabled.

Transferring a next LLI\(_{n+1}\) that updates HPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI\(_n\) trigger.

**Caution:** After a first new trigger, hit\(_{n+1}\) is memorized. If another trigger hit\(_{n+2}\) is detected, and if the hit\(_n\) triggered transfer is still not completed, hit\(_{n+2}\) is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.
The figure below illustrates the trigger hit, memorization, and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

**Figure 126. Trigger hit, memorization and overrun waveform**

**Note:** The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

---

### 13.4.13 HPDMA circular buffering with linked-list programming

**HPDMA circular buffering for memory-toPeripheral and peripheral-to-memory transfers, with a linear addressing channel**

For a circular buffering, with a continuous memory-toPeripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete-transfer event/interrupt generation (HPDMA_CxCR.HTIE = 1 and HPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI1 is needed to restore the memory source (or destination) start address for the memory-toPeripheral transfer (respectively the peripheral-to-memory).

The HPDMA automatically reloads the initially programmed HPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore HPDMA_CxBR1.
The figure below illustrates this programming with a linear addressing HPDMA channel and a source circular buffer.

**Figure 127. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel**

**Note:** With a 2D addressing channel, a single LLI can be used with HPDMA_CxBR1.BRC[10:0] = 1. The user can program a negative memory block address offset with HDMA_CxBR2 and HDMA_CxBR1, in order to jump back to the memory source or destination start address.

If the circular buffering must be executed after some other transfers over the shared HPDMA channel x, the before-last LLI\(_{N-1}\) in the memory is needed to configure the first block transfer. The last LLI\(_N\) restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory).
The figure below illustrates this programming with a linear addressing shared HPDMA channel, and a source circular buffer.

**Figure 128. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel**

### 13.4.14 HPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate an HPDMA block transfer, provided that the allocated channel is implemented with this feature (see **Section 13.3.5**).

If the related HPDMA channel x is also programmed in peripheral flow-control mode (HPDMA_CxTR2.PFREQ = 1):

- The HPDMA block transfer starts as follows:
  - If HPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the HPDMA hardware.
  - If HPDMA_CxBR1.BNDT[15:0] = 0, the HPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.

- The HPDMA block transfer is completed as soon as the first occurrence of any of the following condition occurs:
  - when HPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on HPDMA_CxTR2. Then the software can read the current number of transferred bytes from the source (HPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (HPDMA_CxSAR[31:0] or HPDMA_CxDAR[31:0]).
In peripheral flow-control mode:
- a destination peripheral with an hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- HPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

13.4.15 HPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via HPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:
- An unprivileged read access to a bitfield of this channel is forced to return 0, except for HPDMA_PRIVCFGR that is readable by an unprivileged agent.
- An unprivileged write access to a bitfield of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB/AXI master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the HPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the HPDMA_CxCR.LAP allocated master port.

The HPDMA generates a privileged bus that reflects HPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each HPDMA channel x.

13.4.16 HPDMA error management

The HPDMA can manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as an AHB/AXI single or a burst) to the source or the destination
- The source or destination target reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware). The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1) and the data error (HPDMA_CxSR.DTEF = 1).
- After an HPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After an HPDMA data transfer error, the user must issue a channel reset (set HPDMA_CxCR.RESET) to reset the hardware HPDMA channel data path and the FIFO content, before the user enables again the same channel for a next transfer.
Link transfer error

On a tentative update of a HPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware), the channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1), and the link error (HPDMA_CxSR.ULEF = 1).
- After an HPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (HPDMA_CxLBAR and HPDMA_CxLLR), including the protection attributes.
- After an HPDMA link error, the user must explicitly write the linked-list register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of an HPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (HPDMA_CxCR.EN forced and cleared by the HPDMA hardware), preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1), and a user setting error (HPDMA_CxSR.USEF = 1).
- After an HPDMA user setting error, the user must perform a debug session, taking care of the HPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxBR1.BNDT[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR1.BNDT[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR1.BNDT[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxSAR[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x = 12 to 15, a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxTR3.SAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x = 12 to 15, a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxBR2.BRSAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxDAR[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
– for channel x = 12 to 15, a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxTR3.DAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
– for channel x = 12 to 15, a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR2.BRDAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0])
– a programmed double-word source data width and a programmed AHB source allocated port (HPDMA_CxTR1.SDW_LOG2[1:0] = 11 and HPDMA_CxTR1.SAP = 1)
– a programmed double-word destination data width and a programmed AHB destination allocated port (HPDMA_CxTR1.DDW_LOG2[1:0] = 11 and HPDMA_CxTR1.DAP = 1)
– a programmed linked-list item LLI_{n+1} with a null data transfer (HPDMA_CxLLR.UB1 = 1 and HPDMA_CxBR1.BNDT = 0)

13.4.17 HPDMA autonomous mode

To save dynamic power consumption while the HPDMA executes the programmed linked-list transfers, the HPDMA hardware automatically manages its own clock gating, and generates a clock request output signal to the RCC, whenever the device is in Run, Sleep, or Stop mode, provided that the RCC is programmed with the corresponding HPDMA enable control bits.

For more details about the RCC programming, refer to the RCC section of the reference manual.

For mode details about the availability of the HPDMA autonomous feature versus the device low-power modes, see Section 13.3.2.

The user can program and schedule the execution of a given HPDMA transfer at a LLI_{n} level of an HPDMA channel x, with HPDMA_CxTR2 as follows:

- The software controls and conditions the input of a transfer with TRIGM[1:0], TRIGPOL[1:0], TRIGSEL[5:0], SWREQ, and REQSEL[4:0] for the input trigger and request.
- The software controls and signals the output of a transfer with TCEM[1:0] for generating or not a transfer-complete event, and generating or not an associated half-transfer event).

See HPDMA channel x transfer register 2 (HPDMA_CxTR2) for more details.

When used in low-power modes, this functionality enables a CPU wake-up on a specific transfer completion by the enabled the HPDMA transfer complete interrupt (HPDMA_CxCR.TCIE = 1), or/and enables to continue with the autonomous HPDMA for operating another LLI_{n+1} transfer over the same channel.

The output channel x transfer complete event, hpdma_chx TC, can be programmed as a selected input trigger for a channel if this event is looped-back and connected at HPDMA level (see Section 13.3.6), allowing autonomous and fine HPDMA inter-channel transfer scheduling, without needing a cleared transfer complete flag (TCF).

A given HPDMA channel x asserts its clock request in one of the following conditions:
• if the next transfer to be executed is programmed as conditioned by a trigger (HPDMA_CxTR2.TRIGPOL[1:0] and HPDMA_CxTR2.TRIGM[1:0]), only when the trigger hit occurs
• if the next transfer to be executed is not conditioned by a trigger:
  – if HPDMA_CxTR2.SWREQ = 0, only when the hardware request is asserted by the selected peripheral
  – if HPDMA_CxTR2.SWREQ = 1 (memory-to-memory, GPIO to/from memory), as soon as the HPDMA is enabled

The HPDMA channel x releases its clock request as soon as all the following conditions are met:
• The transfer to be executed is completed.
• The HPDMA channel x is not immediately ready and requested to execute the next transfer.
• If a channel x interrupt was raised, all the flags of the status register that can cause this interrupt, are cleared by a software agent.

When one channel asserts its clock request, the HPDMA asserts its clock request to the RCC. When none channel asserts its clock request, the HPDMA releases its clock request to the RCC.

13.5 HPDMA in debug mode

When the device enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, HPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, HPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

13.6 HPDMA in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. HPDMA interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^1)</td>
<td>The content of HPDMA registers is kept when entering Stop mode. The content of HPDMA registers can be autonomously updated by a next linked-list item from the memory, to perform autonomous data transfers. HPDMA interrupts can cause the device to exit Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The HPDMA is powered down, and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

\(^1\) Refer to Section 13.3.2 to know which Stop mode is supported.
13.7 HPDMA interrupts

There is one HPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 110. HPDMA interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Interrupt enable</th>
<th>Event flag</th>
<th>Event clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPDMA_Chx</td>
<td>Transfer complete</td>
<td>HPDMA_CxCR.TCIE</td>
<td>HPDMA_CxSR.TCF</td>
<td>Writes 1 to HPDMA_CxFCR.TCF</td>
</tr>
<tr>
<td></td>
<td>Half transfer</td>
<td>HPDMA_CxCR.HTIE</td>
<td>HPDMA_CxSR.HTF</td>
<td>Writes 1 to HPDMA_CxFCR.HTF</td>
</tr>
<tr>
<td></td>
<td>Data transfer error</td>
<td>HPDMA_CxCR.DTEIE</td>
<td>HPDMA_CxSR.DTEF</td>
<td>Writes 1 to HPDMA_CxFCR.DTEF</td>
</tr>
<tr>
<td></td>
<td>Update link error</td>
<td>HPDMA_CxCR.ULEIE</td>
<td>HPDMA_CxSR.ULEF</td>
<td>Writes 1 to HPDMA_CxFCR.ULEF</td>
</tr>
<tr>
<td></td>
<td>User setting error</td>
<td>HPDMA_CxCR.USEIE</td>
<td>HPDMA_CxSR.USEF</td>
<td>Writes 1 to HPDMA_CxFCR.USEF</td>
</tr>
<tr>
<td></td>
<td>Suspended</td>
<td>HPDMA_CxCR.SUSPIE</td>
<td>HPDMA_CxSR.SUSPF</td>
<td>Writes 1 to HPDMA_CxFCR.SUSPF</td>
</tr>
<tr>
<td></td>
<td>Trigger overrun</td>
<td>HPDMA_CxCR.TOFIE</td>
<td>HPDMA_CxSR.TOF</td>
<td>Writes 1 to HPDMA_CxFCR.TOF</td>
</tr>
</tbody>
</table>

An HPDMA channel x event can be:
- a transfer complete
- an half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal hpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:
- the corresponding interrupt event xx is enabled (HPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (HPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into HPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by HPDMA_CxTR2.TCEM[1:0] as follows:
- A transfer-complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is an half-block transfer, or a half 2D/repeated block transfer, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].
An half-block transfer occurs when half of the source block size bytes (rounded-up integer of \( \text{HPDMA}_\text{CxBR1.BNDT}[15:0] / 2 \)) is transferred to the destination.

An half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of \( \text{HPDMA}_\text{CxBR1.BRC}[10:0] + 1 / 2 \)) is transferred to the destination.

See **HPDMA channel x transfer register 2 (HPDMA_CxTR2)** for more details.

Note: The interrupt mode must be used (not the polling mode) to be notified on an half transfer when the write data transaction has been completed over the AXI destination allocated port (written at the destination memory-mapped address), and not just before when has been issued, at HPDMA level, this AXI burst transaction.

A transfer error rises in one of the following situations:
- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of an HPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of an HPDMA channel with an unauthorized setting (USEF)

The user must perform a debug session to correct the HPDMA channel programming versus the USEF root causes list (see [Section 13.4.16](#)).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

## 13.8 HPDMA registers

The HPDMA registers must be accessed with an aligned 32-bit word data access.

### 13.8.1 HPDMA privileged configuration register (HPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a HPDMA channel (HPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIV15</td>
<td>PRIV14</td>
<td>PRIV13</td>
<td>PRIV12</td>
<td>PRIV11</td>
<td>PRIV10</td>
<td>PRIV9</td>
<td>PRIV8</td>
<td>PRIV7</td>
<td>PRIV6</td>
<td>PRIV5</td>
<td>PRIV4</td>
<td>PRIV3</td>
<td>PRIV2</td>
<td>PRIV1</td>
<td>PRIV0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.
13.8.2 **HPDMA configuration lock register (HPDMA_RCFGLOCKR)**

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with privileged attributes in order to individually lock, for example at boot time, the privileged attributes of any HPDMA channel/resource (to lock the setting of HPDMA_PRIVCFGR for any channel x at, for example at boot time).

A read access may be privileged or unprivileged.

<table>
<thead>
<tr>
<th>bits 31:16</th>
<th>reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits 15:0</td>
<td>lockx: lock of the configuration of HPDMA_PRIVCFGR.PRIVx until a global HPDMA reset (x = 15 to 0)</td>
</tr>
<tr>
<td>0: privilege configuration of the channel x is writable.</td>
<td></td>
</tr>
<tr>
<td>1: privilege configuration of the channel x is not writable.</td>
<td></td>
</tr>
</tbody>
</table>

13.8.3 **HPDMA masked interrupt status register (HPDMA_MISR)**

Address offset: 0x0C

Reset value: 0x0000 0000

This is a read register.

This register contains the masked interrupt status bit MISx for each channel x. It is a logical OR of all the HPDMA_CxSR flags, each source flag being enabled by the corresponding HPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when writing 1 to the corresponding HPDMA_CxFCR flag clear bit.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel HPDMA_PRIVCFGR.PRIVx. A privileged software can read the full interrupt status. An unprivileged software is restricted to read the status of unprivileged channels, other privileged bitfields returning zero.
13.8.4 HPDMA channel x linked-list base address register (HPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x HPDMA_PRIVCFGR.PRIVx.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the HPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit HPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

Bits 31:16 LBA[31:16]: Linked-list base address of HPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.
13.8.5 HPDMA channel x flag clear register (HPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This is a write register privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFG.RIVx).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag clear
0: No effect
1: Corresponding TOF flag cleared

Bit 13 **SUSPF**: Completed suspension flag clear
0: No effect
1: Corresponding SUSPF flag cleared

Bit 12 **USEF**: User setting error flag clear
0: No effect
1: Corresponding USEF flag cleared

Bit 11 **ULEF**: Update link transfer error flag clear
0: No effect
1: Corresponding ULEF flag cleared

Bit 10 **DTEF**: Data transfer error flag clear
0: No effect
1: Corresponding DTEF flag cleared

Bit 9 **HTF**: Half transfer flag clear
0: No effect
1: Corresponding HTF flag cleared

Bit 8 **TCF**: Transfer complete flag clear
0: No effect
1: Corresponding TCF flag cleared

Bits 7:0 Reserved, must be kept at reset value.
13.8.6 HPDMA channel x status register (HPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is privileged or unprivileged, depending on the privileged state of the channel (HPDMA_PRIVCFGR.PRIVx).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>TOF</td>
<td>SUSPF</td>
<td>USEF</td>
<td>ULEF</td>
<td>DTEF</td>
<td>HTF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
<td>TCF</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **FIFOL[8:0]**: Monitored FIFO level

Number of available write beats in the FIFO, in units of the programmed destination data width (see Section 13.8.8: HPDMA channel x transfer register 1 (HPDMA_CxTR1) DDW_LOG2[1:0], in units of bytes, half-words, words or double-words).

Note: After having suspended an active transfer, the user may need to read FIFOL[8:0], additionally to HPDMA_CxBR1.BDNT[15:0] and HPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (HPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag
0: No trigger overrun event
1: A trigger overrun event occurred.

Bit 13 **SUSPF**: Completed suspension flag
0: No completed suspension event
1: A completed suspension event occurred.

Bit 12 **USEF**: User setting error flag
0: No user setting error event
1: A user setting error event occurred.

Bit 11 **ULEF**: Update link transfer error flag
0: No update link transfer error event
1: A master bus error event occurred while updating a linked-list register from memory.

Bit 10 **DTEF**: Data transfer error flag
0: No data transfer error event
1: A master bus error event occurred on a data transfer.
13.8.7 **HPDMA channel x control register (HPDMA_CxCR)**

Address offset: 0x064 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGx.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>TOIE</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>USEIE</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>ULEIE</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>DTEIE</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>HTIE</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>TCIE</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>PRIO[1:0]</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>LAP</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>LSM</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>SUSP</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>RESET</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>EN</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:22 PRIOR[1:0]: Priority level of the channel x HPDMA transfer versus others
   00: Low priority, low weight
   01: Low priority, mid weight
   10: Low priority, high weight
   11: High priority

   Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 LAP: Linked-list allocated port
   This bit is used to allocate the master port for the update of the HPDMA linked-list registers
   from the memory.
   0: Port 0 (AXI) allocated
   1: Port 1 (AHB) allocated

   Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 LSM: Link step mode
   0: Channel executed for the full linked-list and completed at the end of the last LLI
      (HPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0)
      and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and
      UT3 = UB2 = 0 if present). Then HPDMA_CxBR1.BNDT[15:0] = 0 and
      HPDMA_CxBR1.BRC[10:0] = 0 if present.
   1: Channel executed once for the current LLI
      First the (possible 1D/repeated) block transfer is executed as defined by the current internal
      register file until HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if present.
      Secondly the next linked-list data structure is conditionally uploaded from memory
      as defined by HPDMA_CxLLR. Then channel execution is completed.

   Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 TOIE: Trigger overrun interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled

Bit 13 SUSP: Cmpleted suspension interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled

Bit 12 USE: User setting error interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled

Bit 11 ULE: Update link transfer error interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled

Bit 10 DTE: Data transfer error interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled

Bit 9 HTIE: Half transfer complete interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled

Bit 8 TCIE: Transfer complete interrupt enable
   0: Interrupt disabled
   1: Interrupt enabled
Bits 7:3  Reserved, must be kept at reset value.

Bit 2  **SUSP**: Suspend

Writing 1 to RESET in this register causes the hardware to deassert this SUSP bit, whatever is written into this SUSP. Else:

Software must write 1 in order to suspend an active channel (a channel with an ongoing HPDMA transfer over its master ports).

The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in Figure 110.

0: Write: resume channel, read: channel not suspended
1: Write: suspend channel, read: channel suspended

Bit 1  **RESET**: Reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (HPDMA_CxSR.SUSPF = 1 and HPDMA_CxSR.IDLEF = HPDMA_CxCR.EN = 1)
- channel in disabled state (HPDMA_CxSR.IDLEF = 1 and HPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (HPDMA_CxBR1, HPDMA_CxSAR and HPDMA_CxDAR) before enabling again the channel (see the programming sequence in Figure 111).

0: No channel reset
1: Channel reset

Bit 0  **EN**: Enable

Writing 1 to RESET in this register causes the hardware to deassert this EN bit, whatever is written into this bit 0. Else:

This bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: Write: ignored, read: channel disabled
1: Write: enable channel, read: channel enabled
13.8.8 **HPDMA channel x transfer register 1 (HPDMA_CxTR1)**

Address offset: 0x090 + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of the channel x in HPDMA_PRIVCFGPR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UT1 = 1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>DAP: Destination allocated port</td>
</tr>
<tr>
<td></td>
<td>This bit is used to allocate the master port for the destination transfer</td>
</tr>
<tr>
<td></td>
<td>0: Port 0 (AXI) allocated</td>
</tr>
<tr>
<td></td>
<td>1: Port 1 (AHB) allocated</td>
</tr>
<tr>
<td></td>
<td><em>Note:</em> This bit must be written when EN = 0. This bit is read-only when EN = 1.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>DWX: Destination word exchange</td>
</tr>
<tr>
<td></td>
<td>If the destination data size is not a double-word, this bit is ignored.</td>
</tr>
<tr>
<td></td>
<td>If the destination data size is a double-word and if destination bus is AXI (DAP = 0):</td>
</tr>
<tr>
<td></td>
<td>0: No word-based exchanged within double-word</td>
</tr>
<tr>
<td></td>
<td>1: The two consecutive (post PAM) words are exchanged in each destination double-word.</td>
</tr>
<tr>
<td>27</td>
<td>DHX: Destination half-word exchange</td>
</tr>
<tr>
<td></td>
<td>If the destination data size is shorter than a word, this bit is ignored.</td>
</tr>
<tr>
<td></td>
<td>If the destination data size is a word or double-word and if destination bus is AXI (DAP = 0):</td>
</tr>
<tr>
<td></td>
<td>0: No half-word-based exchanged within word</td>
</tr>
<tr>
<td></td>
<td>1: The two consecutive (post PAM) half-words are exchanged in each destination word.</td>
</tr>
<tr>
<td>26</td>
<td>DBX: Destination byte exchange</td>
</tr>
<tr>
<td></td>
<td>If the destination data size is a byte, this bit is ignored.</td>
</tr>
<tr>
<td></td>
<td>If the destination data size is not a byte:</td>
</tr>
<tr>
<td></td>
<td>0: No byte-based exchange within half-word</td>
</tr>
<tr>
<td></td>
<td>1: The two consecutive (post PAM) bytes are exchanged in each destination half-word.</td>
</tr>
</tbody>
</table>
Bits 25:20 **DBL_1[5:0]**: Destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width **DDW_LOG2[1:0]**.

**Note:** If a burst transfer crossed a 1- or 4-Byte address boundary on respectively an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol.

If the burst length exceeds 16 on an AHB transfer, or if the burst on an AXI transfer is both with fixed addressing (DINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into bursts of lower length, to be compliant with the AHB or AXI protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: Destination incrementing burst

0: Fixed burst
1: Contiguously incremented burst

The destination address, pointed by HPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: Binary logarithm of the destination data width of a burst, in bytes

00: Byte
01: Half-word (2 bytes)
10: Word (4 bytes)
11: If DAP = 0 (AXI), double-word (8 bytes)

if DAP = 1, user setting error reported and no transfer issued

**Note:** A destination burst transfer must have an aligned address with its data width (start address HPDMA_CxDAR[2:0] and address offset HPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see HPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

A destination burst transfer must have an aligned address with its data width (start address HPDMA_CxDAR[2:0] versus DDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

A burst with a double-word data width must be allocated to the AXI master port, else a user setting error is reported and none transfer is issued.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **SAP**: Source allocated port

This bit is used to allocate the master port for the source transfer

0: Port 0 (AXI) allocated
1: Port 1 (AHB) allocated

**Note:** This bit must be written when EN = 0. This bit is read-only when EN = 1.
Bit 13 **SBX**: Source byte exchange within the unaligned half-word of each source word

If the source data width is shorter than a word, this bit is ignored.

If the source data width is a word or a double-word, and if source bus is AXI (SAP = 0):
- 0: No byte-based exchange within the unaligned half-word of each source word
- 1: The two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: Padding/alignment mode

If \(DDW\_LOG2[1:0] = SDW\_LOG2[1:0]\): if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

**Condition: PAM_1**
- 00: Source data is transferred as right aligned, padded with 0s up to the destination data width
- 01: Source data is transferred as right aligned, sign extended up to the destination data width
- 10-11: Successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

**Condition: PAM_2**
- 00: Source data is transferred as right aligned, left-truncated down to the destination data width
- 01: Source data is transferred as left-aligned, right-truncated down to the destination data width
- 10-11: Source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

**Note**: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: Source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If \(SBL\_1[5:0] = 0\), the burst can be named as single. Each data/beat has a width defined by the destination data width \(SDW\_LOG2[1:0]\).

**Note**: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or an AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If the burst length exceeds 16 on an AHB transfer, or if the burst on an AXI transfer is both with fixed addressing \((SINC = 0)\) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: Source incrementing burst

- 0: Fixed burst
- 1: Contiguously incremented burst

The source address, pointed by HPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.
Bits 1:0 **SDW_LOG2[1:0]**: Binary logarithm of the source data width of a burst in bytes

- 00: Byte
- 01: Half-word (2 bytes)
- 10: Word (4 bytes)
- 11: If SAP = 0 (AXI), double-word (8 bytes)
  
  if SAP = 1, user setting error reported and no transfer issued

**Note:** A source block size must be a multiple of the source data width (HPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address HPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

A source burst transfer must have an aligned address with its data width (start address HPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

A burst with a double-word data width must be allocated to the AXI master port, else a user setting error is reported and none transfer is issued.

### 13.8.9 HPDMA channel x transfer register 2 (HPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory, if HPDMA_CxLLR.UT2 = 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Note:** A source block size must be a multiple of the source data width (HPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address HPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

A source burst transfer must have an aligned address with its data width (start address HPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

A burst with a double-word data width must be allocated to the AXI master port, else a user setting error is reported and none transfer is issued.
Bits 31:30  **TCEM[1:0]**: Transfer complete event mode

These bits define the transfer granularity for the transfer complete and half transfer complete events generation.

00: At block level (when HPDMA_CxBR1.BNDT[15:0] = 0): The complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

**Note:** If the initial LLI₀ data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

01: Channel x = 0 to 11, same as 00; channel x = 12 to 15, at 2D/repeated block level (when HPDMA_CxBR1.BRC[10:0] = 0 and HPDMA_CxBR1.BNDT[15:0] = 0), the complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

**Note:** If the initial LLI₀ data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half transfer event is generated.

10: At LLI level: The complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer or a 2D/repeated block transfer for channel x = 12 to 15), if any data transfer.

**Note:** If the initial LLI₀ data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then the half transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI₁.

11: At channel level: The complete transfer event is generated at the end of the last LLI transfer. The half transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address HPDMA_CxLLR.LA[15:2] to zero and clears all the HPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2 if present). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26  Reserved, must be kept at reset value.

Bits 25:24  **TRIGPOL[1:0]**: Trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[5:0].

00: No trigger (masked trigger event)

01: Trigger on the rising edge

10: Trigger on the falling edge

11: Same as 00

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:16  **TRIGSEL[5:0]**: Trigger event input selection

These bits select the trigger event input of the HPDMA transfer (as per Section 13.3.6), with an active trigger event if TRIGPOL[1:0] # 00.
Bits 15:14 **TRIGM[1:0]:** Trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (HPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, an HPDMA transfer is conditioned by at least one trigger hit:
- **00:** at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x = 12 to 15, for each block if a 2D/repeated block is configured with HPDMA_CxBR1.BRC[10:0] ≠ 0).
- **01:** channel x = 0 to 11, same as 00; channel x = 12 to 15, at 2D/repeated block level, the first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.
- **10:** at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.
- **11:** at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.
  - If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.
  - If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The HPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[5:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the HPDMA_CxTR2 with a new value for any of TRIGSEL[5:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_{n} trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_{n} triggered transfer is still not completed, hit_{n+2} is lost and not memorized.

A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

**Note:** When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1/4-Kbyte boundary address crossing maximum burst length versus AHB/AXI protocol): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 **Reserved, must be kept at reset value.**
Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

*Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 13.3.5 for the list of the implemented channels with this feature.)*

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in HPDMA control mode. The HPDMA is programmed with HPDMA_CxCB1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The HPDMA block transfer can be early completed by the peripheral itself (see Section 13.3.5 for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (HPDMA_CxB1.BRC[10:0] must be set to 0 if present)
- the peripheral must be set as the source of the transfer (DREQ = 0).
- data packing to a wider destination width is not supported (if destination width > source data width, HPDMA_CxTR1.PAM[1] must be set to 0).
- HPDMA_CxB1.BNDT[15:0] must be set as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see Section 13.3.4).

Bit 10 **DREQ**: Destination hardware request

This bit is ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: Selected hardware request driven by a source peripheral (request signal taken into account by the HPDMA transfer scheduler over the source/read port)

1: Selected hardware request driven by a destination peripheral (request signal taken into account by the HPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (HPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: Software request

This bit is internally taken into account when HPDMA_CxCR.EN is asserted.

0: No software request. The selected hardware request REQSEL[4:0] is taken into account.

1: Software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[4:0] is ignored.

Bits 8:5 Reserved, must be kept at reset value.
Bits 4:0 **REQSEL[4:0]:** Hardware request selection

These bits are ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per Section 13.3.3.

**Caution:** The user must not assign a same input hardware request (same REQSEL[4:0] value) to different active HPDMA channels (HPDMA_CxCR.EN = 1 and HPDMA_CxTR2.SWREQ = 0 for these channels). The HPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

### 13.8.10 HPDMA channel x block register 1 (HPDMA_CxBR1)

**Address offset:** 0x098 + 0x80 * x (x = 0 to 11)

**Reset value:** 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGPR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfield BNDT[15:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI.
- If HPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

**Bits 31:16** Reserved, must be kept at reset value.
Bits 15:0  **BDNT[15:0]:** Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. **BDNT[15:0]** is programmed in number of bytes, maximum source block size is 64 Kbytes - 1.

Once the last data transfer is completed (**BDNT[15:0] = 0**):
- if HPDMA_CxLLR.UB1 = 1, this bitfield is updated by the LLI in the memory.
- if HPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this bitfield is internally restored to the programmed value.
- if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] = 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
- if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

**Note:** A non-null source block size must be a multiple of the source data width (**BDNT[2:0]** versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (**HPDMA_CxTR1.PAM[1] = 1** and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (**BDNT[2:0]** versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

13.8.11 **HPDMA channel x alternate block register 1 (HPDMA_CxBR1)**

Address offset: 0x098 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:
- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfields **BDNT[15:0]** and **BRC[10:0]**.
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the bitfields **BDNT[15:0]** and **BRC[10:0]** after each execution of this final LLI.
- If HPDMA_CxLLR = 0, **BDNT[15:0]** and **BRC[10:0]** are kept as null, channel x is completed.
**High-performance direct memory access controller (HPDMA)**

<table>
<thead>
<tr>
<th>BRDDEC</th>
<th>BRSDEC</th>
<th>DDEC</th>
<th>SDEC</th>
<th>Res.</th>
<th>BRC[10:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

### Bit 31: BRDDEC
- **Block repeat destination address decrement**
  - 0: At the end of a block transfer, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxBR2.BRDAO to the current HPDMA_CxDAR value (current destination address)
  - 1: At the end of a block transfer, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRDAO from the current HPDMA_CxDAR value (current destination address)

**Note:** On top of this increment/decrement (depending on BRDDEC), HPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the HPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

### Bit 30: BRSDEC
- **Block repeat source address decrement**
  - 0: At the end of a block transfer, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxBR2.BRSAO to the current HPDMA_CxSAR value (current source address)
  - 1: At the end of a block transfer, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRSAO from the current HPDMA_CxSAR value (current source address)

**Note:** On top of this increment/decrement (depending on BRSDEC), HPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the HPDMA_CxTR3.DAO value, as it is done after any programmed burst transfer.

### Bit 29: DDEC
- **Destination address decrement**
  - 0: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)
  - 1: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxTR3.DAO from the current HPDMA_CxDAR value (current destination address)

### Bit 28: SDEC
- **Source address decrement**
  - 0: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)
  - 1: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxTR3.SAO from the current HPDMA_CxSAR value (current source address)

### Bit 27: Reserved
- Must be kept at reset value.
**Bits 26:16**  
**BRC[10:0]: Block repeat counter**

This bitfield contains the number of repetitions of the current block (0 to 2047). When the channel is enabled, this bitfield becomes read-only. After decrements, this bitfield indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

- If HPDMA_CxLLR.UB1 = 1, all HPDMA_CxBR1 bitfields are updated by the next LLI in the memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.
- If all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
- If HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI and data transfer.

**Bits 15:0**  
**BNDT[15:0]: Block number of data bytes to transfer from the source**

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred.

BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1. Once the last data transfer is completed (BNDT[15:0] = 0):

- If HPDMA_CxLLR.UB1 = 1, this bitfield is updated by the LLI in the memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.
- If all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
- If HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

**Note:** A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (HPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.
13.8.12 HPDMA channel x source address register (HPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.USA = 1.

Bits 31:0 \texttt{SA}[31:0]: Source address

This bitfield is the pointer to the address from which the next data is read. During the channel activity, depending on the source addressing mode (HPDMA_CxTR1.SINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.SBL_1[5:0] and HPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by HPDMA_CxBR1.SDEC and HPDMA_CxTR3.SAO[12:0]

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by HPDMA_CxBR1.BRSDEC and HPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (\texttt{SA}[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.
13.8.13  HPDMA channel x destination address register (HPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 15)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGPR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when HPDMA_CxCREN = 0.

This register is read-only when HPDMA_CxCREN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCREN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UDA = 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  DA[31:0]: destination address

This bitfield is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (HPDMA_CxTR1.DINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:
– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.DBL_1[5:0] and HPDMA_CxTR1.DDW_LOG2[1:0]
– the additional destination incremented/decremented offset value as programmed by HPDMA_CxBR1.DDEC and HPDMA_CxTR3.DAO[12:0]
– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 12 to 15), the additional block repeat destination incremented/decremented offset value as programmed by HPDMA_CxBR1.BRDDEC and HPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst [DA[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]]. Else, a user setting error is reported and no transfer is issued.
### 13.8.14 HPDMA channel x transfer register 3 (HPDMA_CxTR3)

Address offset: 0x0A4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFG.PRIVx).

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UT3 = 1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Read</th>
<th>Write</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Bits 31:29** Reserved, must be kept at reset value.

**Bits 28:16** DAO[12:0]: Destination address offset increment

The destination address, pointed by HPDMA_CxDAR, is incremented or decremented (depending on HPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.DINC = 1).

**Note:** A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

**Bits 15:13** Reserved, must be kept at reset value.

**Bits 12:0** SAO[12:0]: Source address offset increment

The source address, pointed by HPDMA_CxSAR, is incremented or decremented (depending on HPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.SINC = 1).

**Note:** A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shortened to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.
13.8.15 HPDMA channel x block register 2 (HPDMA_CxBR2)

Address offset: 0x0A8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UB2 = 1.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRDAO[15:0]</td>
</tr>
<tr>
<td>rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRDDEC) the current destination address (HPDMA_CxDAR) at the end of a block transfer.

*Note:* A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus HPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

**BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).**

<table>
<thead>
<tr>
<th>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRSAO[15:0]</td>
</tr>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRSDEC) the current source address (HPDMA_CxSAR) at the end of a block transfer.

*Note:* A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus HPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

**BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).**
**13.8.16 HPDMA channel x linked-list address register (HPDMA_CxLLR)**

Address offset: 0x0CC + 0x80 * x (x = 0 to 11)
Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGx.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

<table>
<thead>
<tr>
<th>UT1</th>
<th>UT2</th>
<th>UB1</th>
<th>USA</th>
<th>UDA</th>
<th>LA[15:2]</th>
<th>Mem</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory
This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.
0: No HPDMA_CxTR1 update
1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory
This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.
0: No HPDMA_CxTR2 update
1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory
This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory
This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update
Bit 27  **UDA**: Update HPDMA_CxDAR register from memory
   This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
   0: No HPDMA_CxDAR update
   1: HPDMA_CxDAR update

Bits 26:17  Reserved, must be kept at reset value.

Bit 16  **ULL**: Update HPDMA_CxLLR register from memory
   This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
   0: No HPDMA_CxLLR update
   1: HPDMA_CxLLR update

Bits 15:2  **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
   If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
   Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR).
   
   **Note**: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

Bits 1:0  Reserved, must be kept at reset value.
### 13.8.17 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Bit 31  **UT1**: Update HPDMA_CxTR1 from memory
This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.
0: No HPDMA_CxTR1 update
1: HPDMA_CxTR1 update

#### Bit 30  **UT2**: Update HPDMA_CxTR2 from memory
This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.
0: No HPDMA_CxTR2 update
1: HPDMA_CxTR2 update

#### Bit 29  **UB1**: Update HPDMA_CxBR1 from memory
This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: HPDMA_CxBR1 update

#### Bit 28  **USA**: Update HPDMA_CxSAR from memory
This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update
Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
   This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
   0: No HPDMA_CxDAR update
   1: HPDMA_CxDAR update

Bit 26 **UT3**: Update HPDMA_CxTR3 from memory
   This bit controls the update of HPDMA_CxTR3 from the memory during the link transfer.
   0: No HPDMA_CxTR3 update
   1: HPDMA_CxTR3 update

Bit 25 **UB2**: Update HPDMA_CxBR2 from memory
   This bit controls the update of HPDMA_CxBR2 from the memory during the link transfer.
   0: No HPDMA_CxBR2 update
   1: HPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
   This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
   0: No HPDMA_CxLLR update
   1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
   If UT1 = UT2 = UB1 = USA = UDA = ULL= 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
   Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR and HPDMA_CxLLR).
   **Note:** The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

### 13.8.18 HPDMA register map

**Table 111. HPDMA register map and reset values**

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x004  | HPDMA_PRIVCFG |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | HPDMA_RCFGLOC |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | HPDMA_MISR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010 - 0x04C | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.
14 Chrom-GRC (GFXMMU)

14.1 Introduction

The Chrom-GRC (GFXMMU) is a graphical oriented memory management unit aimed to optimize memory usage according to the display shape.

14.2 GFXMMU main features

- Fully programmable display shape to physically store only the visible pixel
- Up to 4 virtual buffers
- Each virtual buffer have 3072 or 4096 bytes per line and 1024 lines
- Each virtual buffer can be physically mapped to any system memory
- Packing and unpacking operation to store 32-bit pixel data into 24-bit packed
- Interrupt in case of buffer overflow (1 per buffer)
- Interrupt in case of memory transfer error

14.3 GFXMMU functional and architectural description

The GFXMMU is responsible of address resolution to convert the virtual buffer address into the physical buffer address.

14.3.1 GFXMMU block diagram

![GFXMMU block diagram](image-url)
14.3.2 GFXMMU internal signals

Table 112. GFXMMU internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gfxmmu_aclk</td>
<td>input</td>
<td>GFXMMU AXI clock</td>
</tr>
<tr>
<td>gfxmmu_hclk</td>
<td>Input</td>
<td>GFXMMU AHB clock</td>
</tr>
<tr>
<td>gfxmmu_it</td>
<td>Output</td>
<td>GFXMMU global interrupt</td>
</tr>
</tbody>
</table>

14.3.3 Virtual memory

The GFXMMU provides a virtual memory space seen by the system masters. This virtual memory space is divided into four virtual buffers.

Virtual buffer

A virtual buffer is seen by any system master as a continuous memory space representing a virtual frame buffer of 1024 lines.

Each line is divided into 256 12-byte or 16-byte blocks depending on the BS bit of the GFXMMU configuration register (GFXMMU_CR).

When the address translation is enabled setting the address translation enable (ATE) bit of the graphic MMU configuration register (GFXMMU_CR), each line can be configured individually to be mapped onto a physical memory.

Depending on the display shape and size, only the necessary blocks are mapped to a physical memory location. This mapping is done programming the LUT entry for each line:

- The enable of the line
- The number of the first “visible” block
- The number of the last “visible” block
- The address offset of the line within the physical buffer

The “visible” blocks can be arranged in the physical buffer in a continuous way programming the address offset of each line.

The LUT is common to all the buffers i.e. all the buffers have the same “shape”.
Virtual buffer overview

For a frame buffer coded in 32 bpp or 16 bpp, the virtual buffer can be configured to have 12-byte or 16-byte blocks. This results in a virtual frame buffer of 768 x 1024 or 1024 x 1024 pixels for 32 bpp and 1536 x 1024 or 2048 x 1024 for 16 bpp.

For a frame buffer coded in 24 bpp, the virtual buffer must be configured to have 12-byte blocks to have an integer number of pixel per line. This results in a virtual frame buffer of 1024 x 1024 pixels for 24 bpp.

Each buffer can be physically mapped anywhere in the physical memory thanks to:

- The physical buffer base address (PBA) field of the GFXMMU buffer x configuration register (GFXMMU_BxCR). It configures the physical location of the 8-Mbyte area where the buffer is mapped.
- The physical buffer location respective to the physical buffer base address is defined by the physical buffer offset (PBO) field of the GFXMMU buffer x configuration register (GFXMMU_BxCR).
The buffer can not overflow the 8-Mbyte boundary of the zone defined by its base address. In case of overflow, the buffer x overflow flag (BxOF) of the GFXMMU status register (GFXMMU_SR) is set and an interrupt is generated if the buffer x overflow interrupt enable (BxOIE) bit of the GFXMMU configuration register (GFXMMU_CR) is set.

**Virtual buffer application use case**

As the physical locations are independently configurable, the four virtual buffers can be physically mapped to non continuous locations. This would allow for example to have the four buffers mapped on to four different SDRAM banks and avoid extra precharge cycles accessing the SDRAM.

As a consequence, one buffer must be used by the CPU/Graphic accelerator for frame buffer calculation while an other one must be used by the LTDC.

The two remaining buffers can be used as a graphical library requiring extra drawing buffers.

**14.3.4 Packing and unpacking**

When addressing 24bpp frame buffers, the virtual buffer must be configured in 12-byte block mode to have an integer number of pixel per block.
Nevertheless, it is possible to access these 24-bit data as if they where unpacked in a 32-bit word.

The packing mode is enabled, per buffer setting the buffer x packing enable (BxPE) bit of the graphic MMU configuration register (GFXMMU_CR).

When the packing mode is enabled and the 12-byte mode is selected, the access to the buffer are done as in 16-byte mode, but either the most significant byte or the less significant byte is ignored to transform each 32-bit word into a 24-bit word.

The ignored byte is selected with the buffer x packing mode (BxPM) bit of the graphic MMU configuration register (GFXMMU_CR).

During write operation MSB or LSB is ignored. During read operation, the ignored MSB or LSB is replaced by the 8-bit default value programmed in the default alpha (DA) field of the graphic MMU default alpha register (GFXMMU_DAR).

The buffer x packing enable (BxPE) bit has no effect when the graphic MMU is configured in 16-byte block.

14.3.5 MMU architecture

The MMU block is responsible of the address resolution. It receives the 24-bit address and returns the physical 23-bit address and a valid signal to indicate the address is physically mapped or not. The MMU also checks overflow of a area boundary.

The MMU LUT is implemented as a 1024 x 35-bit RAM

![MMU block diagram](image)

**Figure 132. MMU block diagram**

**Line block decoder**

The line block decoder is generating the block number and the line number according the address.
Look up RAM

The look up RAM is a 1024 x 35-bit RAM with the following fields:
- 1-bit line enable
- 8-bit first valid block
- 8-bit last valid block
- 18-bit for line offset

As the RAM is bigger than a word, each entry is split into two words on the memory map. The write access are done in two steps:
1. Write the first word with enable/first valid block/last valid block in the GFXMMU_LUTxL memory location (internally buffered).
2. Write the second word with line offset in the GFXMMU_LUTxH memory location (effective write into the memory together with the internally buffered value).

A write in the LUT can happen any time but it can lead to inconsistencies if a master is using the MMU at the same time. As the CPU has the priority during LUT programming, this may slow down MMU calculation.

There is no restriction during read operations, but this may slow down CPU as the MMU has the priority on LUT accesses.

Block validation/comparator

This block is checking if the block is valid.

A block is considered as valid (physically mapped) when:
- Line is enable.
- The block number is greater or equal to the first valid block.
- The block number is lower or equal to the last valid block.

When the block is valid, the physical address generated is considered as correct.

If the result of the MMU evaluation is not valid, the write operations are ignored, and read operations return the default 32-bit value stored in the default value (DV) field of the GFXMMU default value register (GFXMMU_DVR).

Block offset address calculation within the buffer

The block number is added to the line offset to get the offset of the block within the physical buffer.

As a consequence, the line offset stored in the LUT is given by the following formula:

\[
\text{Line offset} = ([\text{Number of visible blocks already used}] - (1^{\text{st}} \text{ visible block}))
\]

with:
- The maximum value for the line offset is when all the block of all the line are used. As the consequence the line offset for the last line can be maximum: \(1023 \times 256 = 0x3:FF00\)
- The minimum value for the line offset is when the last block of the first line is the first valid block: \(-255 = -0xFF\ i.e\ 0x3:FF01\)

As the consequence the full range of the line offset entry of the LUT is used.
Chrom-GRC (GFXMMU)  

Carry is not taken into account as this stage to be able to perform negative offset calculations (values from 0x3:FF01 to 0x3:FFFF)

As the block offset is within a 4-Mbyte buffer, the address generated is 22-bit wide.

**Block offset address calculation**

Once the offset of the block within the buffer as been calculated, this value is added to the offset of the block respective to the physical buffer base address.

The offset of the blocks are defined in registers as shown in *Figure 133*:

![Figure 133. Block validation/comparator implementation](image)

The resulting address and the buffer offset address must be on 23-bit.

The carry is taken into account to trigger address overflow. The carry is propagated to the GFXMMU status register (GFXMMU_S$) to set the buffer x overflow flag (BxOF).

**Example of calculation**

Consider the following configuration for virtual buffer 0:

- 16-byte block mode
- First visible block of line 0: block 7
- Number of visible block in line 0: 10
- First visible block of line 1: block 6
- Number of visible block in line 1: 12
- Address of the physical buffer: 0xC020:0000

The configuration must be:

- The base address of the physical buffer 0: 0xC000:0000
- The offset of buffer 0: 0x20:0000
- First visible block of line 0: block 7
- Last visible block of line 0: block 16
- Block 0 offset of line 0: (0 - 7) = -0x7 = 0x3:FFF9
- First visible block of line 1: block 6
- Last visible block of line 1: block 17
- Block 0 offset of line 1: (10 - 6) = (0xA - 0x6) = 0x4
As a consequence:
- the physical address of block 7 of line 0 is:
  \[0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x7 \text{ without carry}) \times 0x10 = 0xC020:0000\]
- the physical address of block 16 of line 0 is:
  \[0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x10 \text{ without carry}) \times 0x10 = 0xC020:0090\]
- the physical address of block 6 of line 1 is:
  \[0xC000:0000 + 0x20:0000 + (0x4 + 0x6 \text{ without carry}) \times 0x10 = 0xC020:00A0\]
- the physical address of block 17 of line 1 is:
  \[0xC000:0000 + 0x20:0000 + (0x4 + 0x11 \text{ without carry}) \times 0x10 = 0xC020:0150\]

For the same example in 12-byte mode, the user has the same configuration regarding the offsets by the addresses are different in the physical buffer:
- the physical address of block 7 of line 0 is:
  \[0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x7 \text{ without carry}) \times 0x0C = 0xC020:0000\]
- the physical address of block 16 of line 0 is:
  \[0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x10 \text{ without carry}) \times 0x0C = 0xC020:006C\]
- the physical address of block 6 of line 1 is:
  \[0xC000:0000 + 0x20:0000 + (0x4 + 0x6 \text{ without carry}) \times 0x0C = 0xC020:0078\]
- the physical address of block 17 of line 1 is:
  \[0xC000:0000 + 0x20:0000 + (0x4 + 0x11 \text{ without carry}) \times 0x0C = 0xC020:00FC\]

### 14.4 GFXMMU interrupts

An interrupt can be produced on the following events:
- Buffer 0 overflow
- Buffer 1 overflow
- Buffer 2 overflow
- Buffer 3 overflow
- AXI master error

Separate interrupt enable bits are available for flexibility.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buffer 0 overflow</td>
<td>B0OF</td>
<td>B0OIE</td>
</tr>
<tr>
<td>Buffer 1 overflow</td>
<td>B1OF</td>
<td>B1OIE</td>
</tr>
<tr>
<td>Buffer 2 overflow</td>
<td>B2OF</td>
<td>B2OIE</td>
</tr>
<tr>
<td>Buffer 3 overflow</td>
<td>B3OF</td>
<td>B3OIE</td>
</tr>
<tr>
<td>AXI master error</td>
<td>AMEF</td>
<td>AMEIE</td>
</tr>
</tbody>
</table>
# 14.5 GFXMMU registers

## 14.5.1 GFXMMU configuration register (GFXMMU_CR)

Address offset: 0x0000  
Reset value: 0x0000 0000

| Bit 31 | B3PM: Buffer 3 packing mode  
This bit selects the byte to be removed during packing operations on buffer 3  
0: MSB is removed  
1: LSB is removed  

Bit 30 | B3PE: Buffer 3 packing enable  
This bit enables the packing on buffer 3.  
The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.  
0: Packing is disabled  
1: Packing is enable  

Bit 29 | B2PM: Buffer 2 packing mode  
This bit selects the byte to be removed during packing operations on buffer 2  
0: MSB is removed  
1: LSB is removed  

Bit 28 | B2PE: Buffer 2 packing enable  
This bit enables the packing on buffer 2.  
The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.  
0: Packing is disabled  
1: Packing is enable  

Bit 27 | B1PM: Buffer 1 packing mode  
This bit selects the byte to be removed during packing operations on buffer 1  
0: MSB is removed  
1: LSB is removed  

Bit 26 | B1PE: Buffer 1 packing enable  
This bit enables the packing on buffer 1.  
The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.  
0: Packing is disabled  
1: Packing is enable  

Bit 25 | B0PM: Buffer 0 packing mode  
This bit selects the byte to be removed during packing operations on buffer 0  
0: MSB is removed  
1: LSB is removed |
Bit 24  **B0PE**: Buffer 0 packing enable  
This bit enables the packing on buffer 0.  
The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.  
0: Packing is disabled  
1: Packing is enable

Bits 23:16  Reserved, must be kept at reset value.

Bit 15  **ATE**: Address translation enable  
This bit enables the address translation based on the values programmed in the LUT.  
0: Address translation is disable  
1: Address translation is enable

Bits 14:7  Reserved, must be kept at reset value.

Bit 6  **BS**: Block size  
This bit defines the size of the blocks  
0: 16-byte blocks  
1: 12-byte blocks

Bit 5  Reserved, must be kept at reset value.

Bit 4  **AMEIE**: AXI master error interrupt enable  
This bit enables the AXI master error interrupt.  
0: Interrupt disable  
1: Interrupt enabled

Bit 3  **B3OIE**: Buffer 3 overflow interrupt enable  
This bit enables the buffer 3 overflow interrupt.  
0: Interrupt disable  
1: Interrupt enabled

Bit 2  **B2OIE**: Buffer 2 overflow interrupt enable  
This bit enables the buffer 2 overflow interrupt.  
0: Interrupt disable  
1: Interrupt enabled

Bit 1  **B1OIE**: Buffer 1 overflow interrupt enable  
This bit enables the buffer 1 overflow interrupt.  
0: Interrupt disable  
1: Interrupt enabled

Bit 0  **B0OIE**: Buffer 0 overflow interrupt enable  
This bit enables the buffer 0 overflow interrupt.  
0: Interrupt disable  
1: Interrupt enabled
14.5.2 **GFXMMU status register (GFXMMU_SR)**

Address offset: 0x0004
Reset value: 0x0000 0000

![Table showing GFXMMU status register](image)

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

- **Bit 4 AMEF**: AXI master error flag
  - This bit is set when an AXI error happens during a transaction. It is cleared by writing 1 to CAMEF.

- **Bit 3 B3OF**: Buffer 3 overflow flag
  - This bit is set when an overflow occurs during the offset calculation of the buffer 3. It is cleared by writing 1 to CB3OF.

- **Bit 2 B2OF**: Buffer 2 overflow flag
  - This bit is set when an overflow occurs during the offset calculation of the buffer 2. It is cleared by writing 1 to CB2OF.

- **Bit 1 B1OF**: Buffer 1 overflow flag
  - This bit is set when an overflow occurs during the offset calculation of the buffer 1. It is cleared by writing 1 to CB1OF.

- **Bit 0 B0OF**: Buffer 0 overflow flag
  - This bit is set when an overflow occurs during the offset calculation of the buffer 0. It is cleared by writing 1 to CB0OF.

14.5.3 **GFXMMU flag clear register (GFXMMU_FCR)**

Address offset: 0x0008
Reset value: 0x0000 0000

![Table showing GFXMMU flag clear register](image)

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

- **Bit 4 CAMEF**: Clear AXI master error flag
  - Writing 1 clears the AXI master error flag in the GFXMMU_SR register.

- **Bit 3 CB3OF**: Clear buffer 3 overflow flag
  - Writing 1 clears the buffer 3 overflow flag in the GFXMMU_SR register.
Bit 2 **CB2OF**: Clear buffer 2 overflow flag
   Writing 1 clears the buffer 2 overflow flag in the GFXMMU_SR register.

Bit 1 **CB1OF**: Clear buffer 1 overflow flag
   Writing 1 clears the buffer 1 overflow flag in the GFXMMU_SR register.

Bit 0 **CB0OF**: Clear buffer 0 overflow flag
   Writing 1 clears the buffer 0 overflow flag in the GFXMMU_SR register.

### 14.5.4 GFXMMU default value register (GFXMMU_DVR)

- **Address offset**: 0x0010
- **Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:0 DV[31:0]**: Default value
   This field indicates the default 32-bit value which is returned when a master accesses a virtual memory location not physically mapped.

### 14.5.5 GFXMMU default alpha register (GFXMMU_DAR)

- **Address offset**: 0x0014
- **Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:8 DA[7:0]**: Default alpha
   This field indicates the default 8-bit value which is merged with the 24-bit value when a master accesses a virtual memory location in packed mode.
14.5.6 GFXMMU buffer x configuration register (GFXMMU_BxCR)

Address offset: 0x0020 + 0x4 * x, (x = 0 to 3)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:23</td>
<td>PBBA[31:23]: Physical buffer base address. Base address MSB of the physical buffer.</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

14.5.7 GFXMMU LUT entry x low (GFXMMU_LUTxL)

Address offset: 0x1000 + 0x8 * x, (x = 0 to 1023)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 23:16</td>
<td>LVB[7:0]: Last valid block. Number of the last valid block of line number X.</td>
</tr>
<tr>
<td>Bits 15:8</td>
<td>FVB[7:0]: First valid block. Number of the first valid block of line number x.</td>
</tr>
<tr>
<td>Bits 7:1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 0</td>
<td>EN: Enable. Line enable. 0: Line is disabled (no MMU evaluation is performed). 1: Line is enabled (MMU evaluation is performed).</td>
</tr>
</tbody>
</table>
14.5.8  GFXMMU LUT entry x high (GFXMMU_LUTxH)

Address offset: 0x1004 + 0x8 * x, (x = 0 to 1023)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Name</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>GFXMMU_Cr</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0004</td>
<td>GFXMMU_Sr</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0008</td>
<td>GFXMMU_Fcr</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x000c</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0010</td>
<td>GFXMMU_Dvr</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0014</td>
<td>GFXMMU_Dar</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x001c</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

14.5.9  GFXMMU register map

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

Bits 17:0  LO[17:0]: Line offset
Line offset of line number x expressed in number of blocks

Table 114. GFXMMU register map and reset values
Refer to Section 2.3 for the register boundary addresses.
15 Chrom-ART Accelerator controller (DMA2D)

15.1 DMA2D introduction

The Chrom-ART Accelerator (DMA2D) is a specialized DMA dedicated to image manipulation. It can perform the following operations:

- Filling a part or the whole of a destination image with a specific color
- Copying a part or the whole of a source image into a part or the whole of a destination image
- Copying a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- Blending a part and/or two complete source images with different pixel format and copy the result into a part or the whole of a destination image with a different color format

All the classical color coding schemes are supported from 4-bit up to 32-bit per pixel with indexed or direct color mode, including block based YCbCr to handle JPEG decoder output. The DMA2D has its own dedicated memories for CLUTs (color look-up tables).

15.2 DMA2D main features

The main DMA2D features are:

- Single AXI master bus architecture
- AHB slave programming interface supporting 8-, 16-, 32-bit accesses (except for CLUT accesses which are 32-bit)
- User-programmable working area size
- User-programmable offset for sources and destination areas expressed in pixels or bytes
- User-programmable sources and destination addresses on the whole memory space
- Up to two sources with blending operation
- Alpha value that can be modified (source value, fixed value, or modulated value)
- User programmable source and destination color format
- Up to 12 color formats supported from 4-bit up to 32-bit per pixel with indirect or direct color coding
- Block based (8x8) YCbCr support with 4:4:4, 4:2:2 and 4:2:0 chroma sub-sampling factors
- 2 internal memories for CLUT storage in indirect color mode
- Automatic CLUT loading or CLUT programming via the CPU
- User programmable CLUT size
- Internal timer to control AXI bandwidth
- 6 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, memory-to-memory with pixel format conversion and blending, memory-to memory with pixel format conversion, blending and fixed color foreground, and memory-to memory with pixel format conversion, blending and fixed color background.
- Area filling with a fixed color
• Copy from an area to another
• Copy with pixel format conversion between source and destination images
• Copy from two sources with independent color format and blending
• Output buffer byte swapping to support refresh of displays through parallel interface
• Abort and suspend of DMA2D operations
• Watermark interrupt on a user programmable destination line
• Interrupt generation on bus error or access conflict
• Interrupt generation on process completion

15.3 DMA2D functional description

15.3.1 General description

The DMA2D controller performs direct memory transfer. As an AXI master, it can take the control of the AXI bus matrix to initiate AXI transactions.

The DMA2D can operate in the following modes:
• Register-to-memory
• Memory-to-memory
• Memory-to-memory with pixel format conversion
• Memory-to-memory with pixel format conversion and blending
• Memory-to-memory with pixel format conversion, blending and fixed color foreground
• Memory-to-memory with pixel format conversion, blending and fixed color background

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in Figure 134.
15.3.2 DMA2D internal signals

The internal signals of the DMA2D are given in the table below:

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dma2d_aclk</td>
<td>Input</td>
<td>DMA2D AXI clock</td>
</tr>
<tr>
<td>dma2d_hclk</td>
<td>Input</td>
<td>DMA2D AHB clock</td>
</tr>
<tr>
<td>dma2d_it</td>
<td>Output</td>
<td>DMA2D global interrupt request</td>
</tr>
<tr>
<td>dma2d_ctc</td>
<td>Output</td>
<td>DMA2D CLUT transfer complete trigger</td>
</tr>
<tr>
<td>dma2d_tc</td>
<td>Output</td>
<td>DMA2D transfer complete trigger</td>
</tr>
<tr>
<td>dma2d_tw</td>
<td>Output</td>
<td>DMA2D transfer watermark trigger</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trigger name</th>
<th>Direction</th>
<th>Trigger source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>dma2d_ctc</td>
<td>Output</td>
<td>hpdma_trigsel[4]</td>
</tr>
<tr>
<td>dma2d_tc</td>
<td>Output</td>
<td>hpdma_trigsel[5]</td>
</tr>
<tr>
<td>dma2d_tw</td>
<td>Output</td>
<td>hpdma_trigsel[6]</td>
</tr>
</tbody>
</table>
15.3.3 DMA2D control

The DMA2D controller is configured through DMA2D_CR. The user application can perform the following operations:

- Select the operating mode.
- Enable/disable the DMA2D interrupt.
- Start/suspend/abort ongoing data transfers.

15.3.4 DMA2D foreground and background FIFOs

The DMA2D foreground (FG) FG FIFO and background (BG) FIFO fetch the input data to be copied and/or processed.

The FIFOs fetch the pixels according to the color format defined in their respective pixel format converter (PFC).

They are programmed through the following control registers:

- DMA2D foreground memory address register (DMA2D_FGMAR)
- DMA2D foreground offset register (DMA2D_FGOR)
- DMA2D background memory address register (DMA2D_BGMAR)
- DMA2D background offset register (DMA2D_BGBOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR)

When the DMA2D operates in register-to-memory mode, none of the FIFOs is activated.

When the DMA2D operates in memory-to-memory mode (no pixel format conversion nor blending operation), only the FG FIFO is activated, and acts as a buffer.

When the DMA2D operates in memory-to-memory operation with pixel format conversion (no blending operation), the BG FIFO is not activated.

15.3.5 DMA2D foreground and background PFC

DMA2D foreground pixel format converter (PFC) and background pixel format converter perform the pixel format conversion to generate a 32-bit per pixel value. The PFC can also modify the alpha channel.

The first PFC stage converts the color format. The original color format of the foreground and background pixels are configured through CM[3:0] in DMA2D_FGPFCR and DMA2D_BGPFCR, respectively.

The supported input formats are given in the table below.

<table>
<thead>
<tr>
<th>CM[3:0]</th>
<th>Color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>ARGB8888</td>
</tr>
<tr>
<td>0001</td>
<td>RGB888</td>
</tr>
<tr>
<td>0010</td>
<td>RGB565</td>
</tr>
<tr>
<td>0011</td>
<td>ARGB1555</td>
</tr>
<tr>
<td>0100</td>
<td>ARGB4444</td>
</tr>
<tr>
<td>0101</td>
<td>L8</td>
</tr>
</tbody>
</table>

Table 117. Supported color mode in input
The color format are coded as follows:

- **Alpha value field**: transparency
  - 0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- **R field** for Red
- **G field** for Green
- **B field** for Blue
- **L field**: luminance
  - This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format was direct color mode (ARGB/RGB), the extension to 8-bit per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

If the original format is indirect color mode (L/AL), a CLUT is required, and each pixel format converter is associated with a 256 entry 32-bit CLUT.

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and defined in DMA2D_FGCOLR for foreground pixels, and in DMA2D_BGCOLR for background pixels.

The order of the fields in the system memory is defined in the table below.

### Table 117. Supported color mode in input (continued)

<table>
<thead>
<tr>
<th>CM[3:0]</th>
<th>Color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0110</td>
<td>AL44</td>
</tr>
<tr>
<td>0111</td>
<td>AL88</td>
</tr>
<tr>
<td>1000</td>
<td>L4</td>
</tr>
<tr>
<td>1001</td>
<td>A8</td>
</tr>
<tr>
<td>1010</td>
<td>A4</td>
</tr>
<tr>
<td>1011</td>
<td>YCbCr (only for foreground)</td>
</tr>
</tbody>
</table>

The color format are coded as follows:

- **Alpha value field**: transparency
  - 0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- **R field** for Red
- **G field** for Green
- **B field** for Blue
- **L field**: luminance
  - This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format was direct color mode (ARGB/RGB), the extension to 8-bit per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

If the original format is indirect color mode (L/AL), a CLUT is required, and each pixel format converter is associated with a 256 entry 32-bit CLUT.

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and defined in DMA2D_FGCOLR for foreground pixels, and in DMA2D_BGCOLR for background pixels.

The order of the fields in the system memory is defined in the table below.

### Table 118. Data order in memory

<table>
<thead>
<tr>
<th>Color mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGB888</td>
<td>A₃[7:0]</td>
<td>R₃[7:0]</td>
<td>G₃[7:0]</td>
<td>B₀[7:0]</td>
</tr>
<tr>
<td>L8</td>
<td>L₃[7:0]</td>
<td>L₂[7:0]</td>
<td>L₁[7:0]</td>
<td>L₀[7:0]</td>
</tr>
</tbody>
</table>
The 24-bit RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

Once the 32-bit value is generated, the alpha channel can be modified according to AM[1:0] in DMA2D_FGPFCCR or DMA2D_BGPFCCR, as shown in Table 119.

One of the following happens for the alpha channel:
- It is kept as it is (no modification).
- It is replaced by ALPHA[7:0] value in DMA2D_FGPFCCR/DMA2D_BGPFCCR.
- It is replaced by the original alpha value multiplied by the value in DMA2D_FGPFCCR/DMA2D_BGPFCCR divided by 255.

Table 119. Alpha mode configuration

<table>
<thead>
<tr>
<th>AM[1:0]</th>
<th>Alpha mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>No modification</td>
</tr>
<tr>
<td>01</td>
<td>Replaced by value in DMA2D_xxPFCCR</td>
</tr>
<tr>
<td>10</td>
<td>Replaced by original value multiplied by the value in DMA2D_xxPFCCR / 255</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Note: To support the alternate format, the incoming alpha value can be inverted setting AI in DMA2D_FGPFCCR or DMA2D_BGPFCCR. This applies also to the Alpha value stored in the DMA2D_FGPFCCR or DMA2D_BGPFCCR, and in the CLUT.

The R and B fields can also be swapped setting RBS in DMA2D_FGPFCCR or DMA2D_BGPFCCR. This applies also to the RGB order used in the CLUT, and in DMA2D_FGCOLOR or DMA2D_BGCOLOR.

15.3.6 DMA2D foreground and background CLUT interface

The CLUT interface manages the CLUT memory access and the automatic loading of the CLUT.

Three kinds of accesses are possible:
- CLUT read by the PFC during pixel format conversion operation
- CLUT accessed through the AHB slave port when the CPU is reading or writing data into the CLUT
- CLUT written through the AXI master port when an automatic loading of the CLUT is performed

The CLUT memory loading can be done in two different ways:
- Automatic loading
  The following sequence must be followed to load the CLUT:
a) Program the CLUT address into DMA2D_FGCMAR (foreground CLUT) or DMA2D_BGCMAR (background CLUT).

b) Program the CLUT size with CS[7:0] field in DMA2D_FGPFCCR (foreground CLUT), or DMA2D_BGPFCCR (background CLUT).

c) Set START in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict occurs, a CLUT access error interrupt is raised assuming CAEIE = 1 in DMA2D_CR.

- Manual loading

The application has to program the CLUT manually through the DMA2D AHB slave port to which the local CLUT memory is mapped. The foreground CLUT (FGCLUT) is located at address offset 0x0400 and the background CLUT (BGCLUT) at address offset 0x0800.

The CLUT format can be 24 or 32 bits. It is configured through CCM in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT) as shown in table below.

### Table 120. Supported CLUT color mode

<table>
<thead>
<tr>
<th>CCM</th>
<th>CLUT color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>32-bit ARGB8888</td>
</tr>
<tr>
<td>1</td>
<td>24-bit RGB888</td>
</tr>
</tbody>
</table>

The way the CLUT data are organized in the system memory is specified in the table below.

### Table 121. CLUT data order in memory

<table>
<thead>
<tr>
<th>CLUT Color mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGB8888</td>
<td>A0[7:0]</td>
<td>R0[7:0]</td>
<td>G0[7:0]</td>
<td>B0[7:0]</td>
</tr>
<tr>
<td>RGB888</td>
<td>B1[7:0]</td>
<td>R0[7:0]</td>
<td>G0[7:0]</td>
<td>B0[7:0]</td>
</tr>
<tr>
<td></td>
<td>R3[7:0]</td>
<td>G3[7:0]</td>
<td>B3[7:0]</td>
<td>R2[7:0]</td>
</tr>
</tbody>
</table>
15.3.7 DMA2D blender

The DMA2D blender blends the source pixels by pair to compute the resulting pixel.
The blending is performed according to the following equation:

\[
\alpha_{\text{Mult}} = \frac{\alpha_{\text{FG}} \cdot \alpha_{\text{BG}}}{255}
\]

\[
\alpha_{\text{OUT}} = \alpha_{\text{FG}} + \alpha_{\text{BG}} - \alpha_{\text{Mult}}
\]

\[
C_{\text{OUT}} = \frac{C_{\text{FG}} \cdot \alpha_{\text{FG}} + C_{\text{BG}} \cdot \alpha_{\text{BG}} - C_{\text{BG}} \cdot \alpha_{\text{Mult}}}{\alpha_{\text{OUT}}}
\]

with \( C = R \) or \( G \) or \( B \)

Division is rounded to the nearest lower integer

No configuration register is required by the blender. The blender use depends on the DMA2D operating mode defined by \( \text{MODE}[2:0] \) in DMA2D\_CR.

15.3.8 DMA2D output PFC

The output PFC performs the pixel format conversion from 32 bits to the output format defined by \( \text{CM}[2:0] \) in DMA2D\_OPFCCR.

Supported output formats are given in the table below

<table>
<thead>
<tr>
<th>CM[2:0]</th>
<th>Color mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>ARGB8888</td>
</tr>
<tr>
<td>001</td>
<td>RGB888</td>
</tr>
<tr>
<td>010</td>
<td>RGB565</td>
</tr>
<tr>
<td>011</td>
<td>ARGB1555</td>
</tr>
<tr>
<td>100</td>
<td>ARGB4444</td>
</tr>
</tbody>
</table>

Table 122. Supported color mode in output

Note: To support the alternate format, the calculated alpha value can be inverted setting AI bit in DMA2D\_OPFCCR. This applies also to the Alpha value used in DMA2D\_OCOLR. The \( R \) and \( B \) fields can also be swapped setting RBS in DMA2D\_OPFCCR. This applies also to the RGB order used in DMA2D\_OCOLR.

15.3.9 DMA2D output FIFO

The output FIFO programs the pixels according to the color format defined in the output PFC.

The destination area is defined through the following registers:
- DMA2D output memory address register (DMA2D\_OMAR)
- DMA2D output offset register (DMA2D\_OOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D\_NLR)
If the DMA2D operates in register-to-memory mode, the configured output rectangle is filled by the color specified in DMA2D_OCOLR which contains a fixed 32-, 24-, or 16-bit value. The format is selected by CM[2:0] in DMA2D_OPFCCR.

The data are stored into the memory in the order defined in the table below.

### Table 123. Data order in memory

<table>
<thead>
<tr>
<th>Color mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>G₂[7:0]</td>
<td>B₂[7:0]</td>
<td>R₁[7:0]</td>
<td>G₁[7:0]</td>
<td></td>
</tr>
<tr>
<td>R₃[7:0]</td>
<td>G₃[7:0]</td>
<td>B₃[7:0]</td>
<td>R₂[7:0]</td>
<td></td>
</tr>
</tbody>
</table>

The RGB888 aligned on 32 bits is supported through the ARGB8888 mode.

### 15.3.10 DMA2D output FIFO byte reordering

The output FIFO bytes can be reordered to support display frame buffer update through a parallel interface (F(S)MC) directly from the DMA2D.

The reordering of bytes can be done using:
- **RBS** bit to swap red and blue component
- **SB** bit to swap byte two by two in the output FIFO

When the byte swapping is activated (SB = 1 in DMA2D_OPFCR), the number of pixel per line (PL field in DMA2D_NLR) must be even, and the output memory address (MA field in DMA2D_OMAR) must be even. The output line offset computed in bytes (resulting from LOM field in DMA2D_CR and LO field in DMA2D_OOR values) must also be even. If not a configuration error is detected.

### Table 124. Standard data order in memory

<table>
<thead>
<tr>
<th>Color Mode</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>G₂[7:0]</td>
<td>B₂[7:0]</td>
<td>R₁[7:0]</td>
<td>G₁[7:0]</td>
<td></td>
</tr>
<tr>
<td>R₃[7:0]</td>
<td>G₃[7:0]</td>
<td>B₃[7:0]</td>
<td>R₂[7:0]</td>
<td></td>
</tr>
</tbody>
</table>

### 16-bit mode (RGB565)

This mode is supported without byte reordering by the DMA2D.
18/24-bit mode (RGB888)

This mode needs data reordering:
1. The red and the blue have to be swapped (setting the RBS bit).
2. MSB and the LSB bytes of a half-word must be swapped (setting SB bit).

**Table 125. Output FIFO byte reordering steps**

<table>
<thead>
<tr>
<th>Steps</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original data ordering</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G₂[7:0]</td>
<td>B₂[7:0]</td>
<td>R₃[7:0]</td>
<td>G₃[7:0]</td>
<td></td>
</tr>
<tr>
<td>R₃[7:0]</td>
<td>G₃[7:0]</td>
<td>B₃[7:0]</td>
<td>R₃[7:0]</td>
<td></td>
</tr>
<tr>
<td>Setting the RBS bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data ordering after Red and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Setting the SB bit</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R₅[7:0]</td>
<td>B₅[7:0]</td>
<td>G₅[7:0]</td>
<td>R₅[7:0]</td>
<td></td>
</tr>
<tr>
<td>G₅[7:0]</td>
<td>B₅[7:0]</td>
<td>R₅[7:0]</td>
<td>G₅[7:0]</td>
<td></td>
</tr>
<tr>
<td>B₅[7:0]</td>
<td>G₅[7:0]</td>
<td>R₅[7:0]</td>
<td>B₅[7:0]</td>
<td></td>
</tr>
</tbody>
</table>
15.3.11 DMA2D AXI master port timer

An 8-bit timer is embedded into the AXI master port to provide an optional limitation of the bandwidth on the crossbar. This timer is clocked by the AXI clock and counts a dead time between two consecutive accesses. This limits the bandwidth usage.

The timer enabling and dead time value are configured through DMA2D_AMPTCR.

15.3.12 DMA2D transactions

DMA2D transactions consist of a sequence of a given number of data transfers. Number of data and width can be programmed by software.

Each DMA2D data transfer is composed of up to four steps:
1. Data loading from the memory location pointed by DMA2D_FGMAR and pixel format conversion as defined in DMA2D_FGCR
2. Data loading from a memory location pointed by DMA2D_BGMAR and pixel format conversion as defined in DMA2D_BGCR
3. Blending of all retrieved pixels according to the alpha channels resulting of PFC operation on alpha values
4. Pixel format conversion of resulting pixels according to DMA2D_OCR and programming of data to the memory location addressed through DMA2D_OMAR

15.3.13 DMA2D configuration

Both source and destination data transfers can target peripherals and memories in the whole 4-Byte memory area, at addresses ranging between 0x0000 0000 and 0xFFFF FFFF.

The DMA2D can operate in any of the following modes selected through MODE[2:0] in DMA2D_CR: 
- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending
- Memory-to-memory with PFC, blending and fixed FG color
- Memory-to-memory with PFC, blending and fixed BG color

Register-to-memory

The register-to-memory mode is used to fill a user defined area with a predefined color. The color format is set in DMA2D_OPFCCR.

Table 125. Output FIFO byte reordering steps (continued)

<table>
<thead>
<tr>
<th>Steps</th>
<th>@ + 3</th>
<th>@ + 2</th>
<th>@ + 1</th>
<th>@ + 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data ordering after byte swapping (SB set)</td>
<td>B3[7:0]</td>
<td>R1[7:0]</td>
<td>R3[7:0]</td>
<td>G0[7:0]</td>
</tr>
<tr>
<td></td>
<td>G3[7:0]</td>
<td>B3[7:0]</td>
<td>B2[7:0]</td>
<td>R3[7:0]</td>
</tr>
</tbody>
</table>
The DMA2D does not perform any data fetching from any source. It just writes the color defined in DMA2D_OCOLR to the area located at the address pointed by DMA2D_OMAR and defined in DMA2D_NLR and DMA2D_OOR.

**Memory-to-memory**

In memory-to-memory mode, the DMA2D does not perform any graphical data transformation. The foreground input FIFO acts as a buffer, and data are transferred from the source memory location defined in DMA2D_FGMAR, to the destination memory location pointed by DMA2D_OMAR.

The color mode programmed by CM[3:0] in DMA2D_FGPFCR defines the number of bits per pixel for both input and output.

The size of the area to be transferred is defined by DMA2D_NLR and DMA2D_FGOR for the source, and by DMA2D_NLR and DMA2D_OOR for the destination.

**Memory-to-memory with PFC**

In this mode, the DMA2D performs a pixel format conversion of the source data, and stores them in the destination memory location.

The size of the areas to be transferred are defined by DMA2D_NLR and DMA2D_FGOR for the source, and by DMA2D_NLR and DMA2D_OOR for the destination.

Data are fetched from the location defined in DMA2D_FGMAR, and processed by the foreground PFC. The original pixel format is configured through DMA2D_FGPFCR.

If the original pixel format is direct-color mode, the color channels are all expanded to 8 bits.

If the pixel format is indirect color mode, the associated CLUT has to be loaded into the CLUT memory.

The CLUT loading can be done automatically by following the sequence below:

1. Set the CLUT address into DMA2D_FGCMAR.
2. Set the CLUT size with CS[7:0] in DMA2D_FGPFCR.
3. Set the CLUT format (24 or 32 bits) with CCM in DMA2D_FGPFCR.
4. Start the CLUT loading by setting START in DMA2D_FGPFCR.

Once the CLUT loading is complete, CTCIF flag in DMA2D_IFR is raised, and an interrupt is generated if CTCIE is set in DMA2D_CR. The automatic CLUT loading process cannot work in parallel with classical DMA2D transfers.

The CLUT can also be filled by the CPU or by any other master through the AHB port. The access to the CLUT is not possible when a DMA2D transfer is ongoing, and uses the CLUT (indirect color format).

In parallel to the color conversion process, the alpha value can be added or changed depending on the value programmed in DMA2D_FGPFCR. If the original image does not have an alpha channel, a default alpha value of 0xFF is automatically added to obtain a fully opaque pixel.

The alpha value is modified as follows according to AM[1:0] in DMA2D_FGPFCR:

- It is unchanged.
- It is replaced by ALPHA[7:0] value in DMA2D_FGPFCR.
- It is replaced by the original value multiplied by ALPHA[7:0] value in DMA2D_FGPFCR divided by 255.
The resulting 32-bit data are encoded by the output PFC into the format specified by CM[2:0] in DMA2D_OPFCCR. The output pixel format cannot be the indirect mode since no CLUT generation process is supported.

Processed data are written into destination memory location pointed by DMA2D_OMAR.

**Memory-to-memory with PFC and blending**

In this mode, two sources are fetched in the foreground and background FIFOs from the memory locations defined by DMA2D_FGMAR and DMA2D_BGMAR respectively.

The two pixel format converters have to be configured as described in memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory.

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended according to the equation below:

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}\alpha_{FG} + C_{BG}\alpha_{BG} - C_{BG}\alpha_{Mult}}{\alpha_{OUT}}$$

with $\alpha = \text{R or G or B}$

Division are rounded to the nearest lower integer

The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and data are written into the destination memory location pointed by DMA2D_OMAR.

**Memory-to-memory with PFC, blending and fixed color FG**

In this mode, only one source is fetched in the background FIFO from the memory location defined by DMA2D_BGMAR.

The value of the foreground color is given by DMA2D_FGCOLR, and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to AM[1:0] and ALPHA[7:0] in DMA2D_FGPFCCR.

The two pixel format converters have to be configured as described in memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory.

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended together. The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format. Data are written into the destination memory location pointed by DMA2D_OMAR.

**Memory-to-memory with PFC, blending and fixed color BG**

In this mode, only open source is fetched in the foreground FIFO from the memory location defined by DMA2D_FGMAR.
The value of the background color is given by DMA2D_BGCOLR, and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to AM[1:0] and ALPHA[7:0] in DMA2D_BGPFCR.

The two pixel format converters have to be configured as described in memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory.

Once each pixel has been converted into 32 bits by their respective PFCs, all pixels are blended together. The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format. Data are written into the destination memory location pointed by DMA2D_OMAR.

Configuration error detection

The DMA2D checks that the configuration is correct before any transfer. The configuration error interrupt flag is set by hardware when a wrong configuration is detected when a new transfer/automatic loading starts. An interrupt is then generated if CEIE = 1 in DMA2D_CR.

The wrong configurations that can be detected are listed below:

- Foreground CLUT automatic loading: MA bits in DMA2D_FGCMAR are not aligned with CCM in DMA2D_FGPFCR.
- Background CLUT automatic loading: MA bits in DMA2D_BGCMAR are not aligned with CCM in DMA2D_BGPFCR.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): MA bits in DMA2D_FGMAR are not aligned with CM in DMA2D_FGPFCR.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): CM bits in DMA2D_FGPFCR are invalid.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): PL bits in DMA2D_NLR are odd while CM is DMA2D_FGPFCR is A4 or L4.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): LO bits in DMA2D_FGOR are odd while CM in DMA2D_FGPFCR is A4 or L4, and LOM in DMA2D_CR is pixel mode.
- Memory transfer (only in blending mode, except in memory-to-memory with blending and fixed color FG): MA bits in DMA2D_BGMAR are not aligned with the CM in DMA2D_BGPFCR.
- Memory transfer (only in blending and blending with fixed color FG): CM bits in DMA2D_BGPFCR are invalid.
- Memory transfer (only in blending and blending with fixed color FG mode): PL bits in DMA2D_NLR odd while CM in DMA2D_BGPFCR is A4 or L4.
- Memory transfer (only in blending and blending with fixed color FG): LO bits in DMA2D_BGOR are odd while CM in DMA2D_BGPFCR is A4 or L4, and LOM in DMA2D_CR is pixel mode.
- Memory transfer (except in memory-to-memory): MA bits in DMA2D_OMAR are not aligned with CM in DMA2D_OPFCCR.
- Memory transfer (except in memory-to-memory): CM bits in DMA2D_OPFCCR are invalid.
• Memory transfer with byte swapping: PL bits in DMA2D_NLR are odd, or MA bits in DMA2D_OMAR are odd, or LO in bytes (resulting from LOM in DMA2D_CR and LO in DMA2D_OOR) are odd while SB = 1 in DMA2D_OPFCCR.

• Memory transfer: NL bits in DMA2D_NLR are all zeros.

• Memory transfer: PL bits in DMA2D_NLR are all zeros.

• Memory transfer: MODE bits in DMA2D_CR are invalid.

• YCbCr format: when a CLUT loading starts setting START in DMA2D_FGPFCCR

• YCbCr format: when memory-to-memory mode is selected

• YCbCr format: when YCbCr4:4:4 is selected, and the sum of number of pixel (PL) and line offset LO is not a multiple of 8 pixels

• YCbCr format: when YCbCr4:2:2 or YCbCr4:2:0 is selected, and the sum of number of pixel (PL) and line offset LO is not a multiple of 16 pixels

15.3.14 YCbCr support

The DMA2D foreground plane can support 8x8 block-based YCbCr as output by the JPEG decoder with different chroma subsampling factors.

The memory organization follows the standard JFIF rules:
• Each of the three color components must be coded on 8-bit.
• Each component must be arranged by blocks of 8x8 (64 bytes) called MCU.

Depending of the chroma subsampling factor, the MCU must be arranged in the memory as described in the table below.

<table>
<thead>
<tr>
<th>Sub-sampling</th>
<th>@</th>
<th>@ + 64</th>
<th>@ + 128</th>
<th>@ +192</th>
<th>@+256</th>
<th>@ + 320</th>
</tr>
</thead>
<tbody>
<tr>
<td>4:4:4</td>
<td>Y1</td>
<td>Cb1</td>
<td>Cr1</td>
<td>Y2</td>
<td>Cb2</td>
<td>Cr2</td>
</tr>
<tr>
<td>4:2:2</td>
<td>Y1</td>
<td>Y2</td>
<td>Cb12</td>
<td>Cr12</td>
<td>Y3</td>
<td>Y4</td>
</tr>
<tr>
<td>4:2:0</td>
<td>Y1</td>
<td>Y2</td>
<td>Y3</td>
<td>Y4</td>
<td>Cb1234</td>
<td>Cr1234</td>
</tr>
</tbody>
</table>

The chroma subsampling factor is configured through CSS in DMA2D_FGPFCCR.

Once the DMA2D has started with the foreground configured in YCbCr color mode, the first two chroma MCUs are loaded in the foreground CLUT. Once the chroma MCUs are loaded, the DMA2D performs the loading of the Y MCU as for a classical color mode.

15.3.15 DMA2D transfer control (start, suspend, abort, and completion)

Once the DMA2D is configured, the transfer can be launched by setting START in DMA2D_CR. Once the transfer is completed, START is automatically reset, and TCIF flag is raised in DMA2D_ISR. An interrupt can be generated if TCIE is set in DMA2D_CR.

The user application can suspend the DMA2D at any time by setting SUSP in DMA2D_CR. The transaction can be aborted by setting ABORT in DMA2D_CR, or can be restarted by resetting SUSP in DMA2D_CR.

The user application can abort at any time an ongoing transaction by setting ABORT in DMA2D_CR. In this case, the TCIF flag is not raised.
Automatic CLUT transfers can also be aborted or suspended by using their own START bits in DMA2D_FGPFCCR and DMA2D_BGPFCCR.

### 15.3.16 Watermark

A watermark can be programmed to generate an interrupt when the last pixel of a given line has been written to the destination memory area.

The line number is defined in LW[15:0] in DMA2D_LWR.

When the last pixel of this line has been transferred, TWIF flag is raised in DMA2D_ISR, and an interrupt is generated if TWIE is set in DMA2D_CR.

### 15.3.17 Error management

Two kinds of errors can be triggered:

- AXI master port errors signaled by TEIF flag in DMA2D_ISR
- Conflicts caused by CLUT access (CPU trying to access the CLUT while a CLUT loading or a DMA2D transfer is ongoing) signaled by CAEIF flag in DMA2D_ISR

Both flags are associated to their own interrupt enable flag in DMA2D_CR to generate an interrupt if need be (TEIE and CAEIE).

### 15.3.18 AXI dead time

To limit the AXI bandwidth use, a dead time between two consecutive AXI accesses can be programmed. This feature can be enabled by setting EN in DMA2D_AMTCR.

The dead-time value is stored in DT[7:0] in DMA2D_AMTCR. This value represents the guaranteed minimum number of cycles between two consecutive transactions on AXI bus.

The update of the dead-time value while the DMA2D is running is taken into account for the next AXI transfer.

### 15.4 DMA2D interrupts

An interrupt can be generated on the following events:

- Configuration error
- CLUT transfer complete
- CLUT access error
- Transfer watermark reached
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration error</td>
<td>CEIF</td>
<td>CEIE</td>
</tr>
<tr>
<td>CLUT transfer complete</td>
<td>CTCIF</td>
<td>CTCIE</td>
</tr>
</tbody>
</table>
15.5 DMA2D registers

15.5.1 DMA2D control register (DMA2D_CR)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLUT access error</td>
<td>CAEIF</td>
<td>CAEIE</td>
</tr>
<tr>
<td>Transfer watermark</td>
<td>TWF</td>
<td>TWIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCIF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEIF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>

Table 127. DMA2D interrupt requests (continued)

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE[2:0]:** DMA2D mode

This bit is set and cleared by software. It cannot be modified while a transfer is ongoing.

000: Memory-to-memory (FG fetch only)
001: Memory-to-memory with PFC (FG fetch only with FG PFC active)
010: Memory-to-memory with blending (FG and BG fetch with PFC and blending)
011: Register-to-memory (no FG nor BG, only output stage active)
100: Memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active)
101: Memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active)
Others: Reserved

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

Bit 13 **CEIE:** Configuration error (CE) interrupt enable

This bit is set and cleared by software.
0: CE interrupt disabled
1: CE interrupt enabled

Bit 12 **CTCIE:** CLUT transfer complete (CTC) interrupt enable

This bit is set and cleared by software.
0: CTC interrupt disabled
1: CTC interrupt enabled
Bit 11 **CAEIE**: CLUT access error (CAE) interrupt enable  
This bit is set and cleared by software.  
0: CAE interrupt disabled  
1: CAE interrupt enabled

Bit 10 **TWIE**: Transfer watermark (TW) interrupt enable  
This bit is set and cleared by software.  
0: TW interrupt disabled  
1: TW interrupt enabled

Bit 9 **TCIE**: Transfer complete (TC) interrupt enable  
This bit is set and cleared by software.  
0: TC interrupt disabled  
1: TC interrupt enabled

Bit 8 **TEIE**: Transfer error (TE) interrupt enable  
This bit is set and cleared by software.  
0: TE interrupt disabled  
1: TE interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LOM**: Line offset mode  
This bit configures how the line offset is expressed (pixels or bytes) for the foreground, background and output.  
This bit is set and cleared by software. It can not be modified while a transfer is ongoing.  
0: Line offsets expressed in pixels  
1: Line offsets expressed in bytes

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **ABORT**: Abort  
This bit can be used to abort the current transfer. This bit is set by software, and is automatically reset by hardware when START = 0.  
0: No transfer abort requested  
1: Transfer abort requested

Bit 1 **SUSP**: Suspend  
This bit can be used to suspend the current transfer. This bit is set and reset by software. It is automatically reset by hardware when START = 0.  
0: Transfer not suspended  
1: Transfer suspended

Bit 0 **START**: Start  
This bit can be used to launch the DMA2D according to parameters loaded in the various configuration registers. This bit is automatically reset by the following events:  
– at the end of the transfer  
– when the data transfer is aborted by the user by setting ABORT in this register  
– when a data transfer error occurs  
– when the data transfer has not started due to a configuration error, or another transfer operation already ongoing (automatic CLUT loading)
15.5.2 DMA2D interrupt status register (DMA2D_ISR)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CEIF</td>
<td>CTCIF</td>
<td>CAEIF</td>
<td>TWIF</td>
<td>TCIF</td>
<td>TEIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 5 **CEIF**: Configuration error interrupt flag
This bit is set when START is set in DMA2D_CR, DMA2DFGPFCCR or DMA2D_BGPFCR, and a wrong configuration has been programmed.

Bit 4 **CTCIF**: CLUT transfer complete interrupt flag
This bit is set when the CLUT copy from a system memory area to the internal DMA2D memory is complete.

Bit 3 **CAEIF**: CLUT access error interrupt flag
This bit is set when the CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal DMA2D.

Bit 2 **TWIF**: Transfer watermark interrupt flag
This bit is set when the last pixel of the watermarked line has been transferred.

Bit 1 **TCIF**: Transfer complete interrupt flag
This bit is set when a DMA2D transfer operation is complete (data transfer only).

Bit 0 **TEIF**: Transfer error interrupt flag
This bit is set when an error occurs during a DMA transfer (data transfer or automatic CLUT loading).

15.5.3 DMA2D interrupt flag clear register (DMA2D_IFCR)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CCEIF</td>
<td>CTCIF</td>
<td>CAECIF</td>
<td>CTWIF</td>
<td>CTCIF</td>
<td>CTEIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 |

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

Bit 5 **CCEIF**: Clear configuration error interrupt flag
Programming this bit to 1 clears the CEIF flag in DMA2D_ISR.
Bit 4 **CCTCIF**: Clear CLUT transfer complete interrupt flag  
Programming this bit to 1 clears the CTCIF flag in DMA2D_ISR.

Bit 3 **CAECIF**: Clear CLUT access error interrupt flag  
Programming this bit to 1 clears the CAEIF flag in DMA2D_ISR.

Bit 2 **CTWIF**: Clear transfer watermark interrupt flag  
Programming this bit to 1 clears the TWIF flag in DMA2D_ISR.

Bit 1 **CTCIF**: Clear transfer complete interrupt flag  
Programming this bit to 1 clears the TCIF flag in DMA2D_ISR.

Bit 0 **CTEIF**: Clear transfer error interrupt flag  
Programming this bit to 1 clears the TEIF flag in DMA2D_ISR.

### 15.5.4 DMA2D foreground memory address register (DMA2D_FGMAR)

Address offset: 0x00C  
Reset value: 0x0000 0000  
This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

MA[31:16]
```

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

MA[15:0]
```

Bits 31:0 **MA[31:0]**: Memory address, address of the data used for the foreground image  
The address alignment must match the image format selected: a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned, and a 4-bit per pixel format must be 8-bit aligned.

### 15.5.5 DMA2D foreground offset register (DMA2D_FGOR)

Address offset: 0x010  
Reset value: 0x0000 0000  
This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

LO[15:0]
```

Bits 31:16 Reserved, must be kept at reset value.
15.5.6 DMA2D background memory address register (DMA2D_BGMAR)

Address offset: 0x014
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

Bits 15:0 LO[15:0]: Line offset
This field gives the line offset used for the foreground image, expressed:
– in pixels when LOM = 0 in DMA2D_CR. Only LO[13:0] bits are considered, LO[15:14] bits are ignored.
– in bytes when LOM = 1
This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.
If the image format is 4-bit per pixel, the line offset must be even.

Bits 31:0 MA[31:0]: Memory address, address of the data used for the background image
The address alignment must match the image format selected: a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

15.5.7 DMA2D background offset register (DMA2D_BGOR)

Address offset: 0x018
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

Bits 31:16 Reserved, must be kept at reset value.
15.5.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR)

Address offset: 0x01C
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

<table>
<thead>
<tr>
<th>Address</th>
<th>Alpha[7:0]</th>
<th>RBS</th>
<th>AI</th>
<th>CSS[1:0]</th>
<th>AM[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:24</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bits 23:22</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bits 21</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bits 20</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Bits 19:18</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 **ALPHA[7:0]:** Alpha value
These bits define a fixed alpha channel value which can replace the original alpha value, or be multiplied by the original alpha value, according to the alpha mode selected through AM[1:0] in this register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS:** Red/Blue swap
This bit allows to swap Red and Blue to support BGR or ABGR color formats.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI:** Alpha inverted
This bit inverts the alpha value.
0: Regular alpha
1: Inverted alpha

Bits 19:18 **CSS[1:0]:** Chroma subsampling
These bits define the chroma subsampling mode for YCbCr color mode.
00: 4:4:4 (no chroma subsampling)
01: 4:2:2
10: 4:2:0
Others: Reserved
Bits 17:16  AM[1:0]: Alpha mode
   These bits select the alpha channel value to be used for the foreground image.
   00: No modification of the foreground image alpha channel value
   01: Replace original foreground image alpha channel value by ALPHA[7:0]
   10: Replace original foreground image alpha channel value by ALPHA[7:0] multiplied with
       original alpha channel value
   Others: Reserved

Bits 15:8  CS[7:0]: CLUT size
   These bits define the size of the CLUT used for the foreground image.
   The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6  Reserved, must be kept at reset value.

Bit 5  START: Start
   This bit can be set to start the automatic loading of the CLUT. It is automatically reset:
   – at the end of the transfer
   – when the transfer is aborted by the user by setting ABORT in DMA2D_CR
   – when a transfer error occurs
   – when the transfer has not started due to a configuration error or another transfer
     operation already ongoing (data transfer or automatic background CLUT transfer)

Bit 4  CCM: CLUT color mode
   This bit defines the color format of the CLUT.
   0: ARGB8888
   1: RGB888

Bits 3:0  CM[3:0]: Color mode
   These bits define the color format of the foreground image.
   0000: ARGB8888
   0001: RGB888
   0010: RGB565
   0011: ARGB1555
   0100: ARGB4444
   0101: L8
   0110: AL44
   0111: AL88
   1000: L4
   1001: A8
   1010: A4
   1011: YCbCr
   Others: Reserved
15.5.9 DMA2D foreground color register (DMA2D_FGCOLR)

Address offset: 0x020
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RED[7:0]</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

GREEN[7:0] BLUE[7:0]

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 RED[7:0]: Red value for the A4 or A8 mode of the foreground image
Used also for fixed color FG in memory-to-memory mode with blending and fixed color FG (BG fetch only with FG and BG PFC active).

Bits 15:8 GREEN[7:0]: Green value for the A4 or A8 mode of the foreground image
Used also for fixed color FG in memory-to-memory mode with blending and fixed color FG (BG fetch only with FG and BG PFC active).

Bits 7:0 BLUE[7:0]: Blue value for the A4 or A8 mode of the foreground image
Used also for fixed color FG in memory-to-memory mode with blending and fixed color FG (BG fetch only with FG and BG PFC active).

15.5.10 DMA2D background PFC control register (DMA2D_BGPFCCR)

Address offset: 0x024
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ALPHA[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RW</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

CS[7:0] START CCM CM[3:0]

Bits 31:24 ALPHA[7:0]: Alpha value
These bits define a fixed alpha channel value which can replace the original alpha value, or be multiplied with the original alpha value according to the alpha mode selected with AM[1:0].

Bits 23:22 Reserved, must be kept at reset value.
Bit 21  **RBS**: Red/Blue swap  
This bit allows to swap Red and Blue to support BGR or ABGR color formats.  
0: Regular mode (RGB or ARGB)  
1: Swap mode (BGR or ABGR)  

Bit 20  **AI**: Alpha Inverted  
This bit inverts the alpha value.  
0: Regular alpha  
1: Inverted alpha  

Bits 19:18  Reserved, must be kept at reset value.  

Bits 17:16  **AM[1:0]**: Alpha mode  
These bits define which alpha channel value to be used for the background image.  
00: No modification of the foreground image alpha channel value  
01: Replace original background image alpha channel value by ALPHA[7:0]  
10: Replace original background image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value  
Others: Reserved  

Bits 15:8  **CS[7:0]**: CLUT size  
These bits define the size of the CLUT used for the BG.  
The number of CLUT entries is equal to CS[7:0] + 1.  

Bits 7:6  Reserved, must be kept at reset value.  

Bit 5  **START**: Start  
This bit is set to start the automatic loading of the CLUT. This bit is automatically reset:  
– at the end of the transfer  
– when the transfer is aborted by the user by setting ABORT bit in DMA2D_CR  
– when a transfer error occurs  
– when the transfer has not started due to a configuration error or another transfer operation already on going (data transfer or automatic background CLUT transfer)  

Bit 4  **CCM**: CLUT color mode  
These bits define the color format of the CLUT.  
0: ARGB8888  
1: RGB888  

Bits 3:0  **CM[3:0]**: Color mode  
These bits define the color format of the foreground image.  
0000: ARGB8888  
0001: RGB888  
0010: RGB565  
0011: ARGB1555  
0100: ARGB4444  
0101: L8  
0110: AL44  
0111: AL88  
1000: L4  
1001: A8  
1010: A4  
Others: Reserved
15.5.11 DMA2D background color register (DMA2D_BGCOLR)

Address offset: 0x028
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RAM[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:16 **RED[7:0]**: Red value for the A4 or A8 mode of the background
Used also for fixed color BG in memory-to-memory mode with blending and fixed color BG (FG fetch only with FG and BG PFC active).

Bits 15:8 **GREEN[7:0]**: Green value for the A4 or A8 mode of the background
Used also for fixed color BG in memory-to-memory mode with blending and fixed color BG (FG fetch only with FG and BG PFC active).

Bits 7:0 **BLUE[7:0]**: Blue value for the A4 or A8 mode of the background
Used also for fixed color BG in memory-to-memory mode with blending and fixed color BG (FG fetch only with FG and BG PFC active).

15.5.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)

Address offset: 0x02C
Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RAM[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **MA[31:0]**: Memory address
Address of the data used for the CLUT address dedicated to the foreground image.
If the foreground CLUT format is 32-bit, the address must be 32-bit aligned.
15.5.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR)

Address offset: 0x030
Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

Bits 31:0 **MA[31:0]**: Memory address
Address of the data used for the CLUT address dedicated to the background image.
If the background CLUT format is 32-bit, the address must be 32-bit aligned.

15.5.14 DMA2D output PFC control register (DMA2D_OPFCCR)

Address offset: 0x034
Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

Bit 21 **RBS**: Red/Blue swap
This bit allows to swap Red and Blue to support BGR or ABGR color formats.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted
This bit inverts the alpha value.
0: Regular alpha
1: Inverted alpha

Bits 19:9 Reserved, must be kept at reset value.
Bit 8  **SB**: Swap bytes  
When this bit is set, the bytes in the output FIFO are swapped two by two. The number of pixels per line (PL) must be even, and the output memory address (OMAR) must be even.  
0: Bytes in regular order in the output FIFO  
1: Bytes swapped two by two in the output FIFO  

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

Bits 2:0  **CM[2:0]**: Color mode  
These bits define the color format of the output image.  
000: ARGB8888  
001: RGB888  
010: RGB565  
011: ARG8155  
100: ARGB4444  
Others: Reserved  

### 15.5.15 DMA2D output color register (DMA2D_OCOLR)

Address offset: 0x038  
Reset value: 0x0000 0000  
The same register is used to show the color values, with different formats depending on the color mode.  
This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.  

**ARGB8888 or RGB888 color mode**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ALPHA[7:0]**  
Bits 31:24  **ALPHA[7:0]**: Alpha channel value of the output color in ARGB8888 mode (otherwise reserved)  

**RED[7:0]**  
Bits 23:16  **RED[7:0]**: Red value of the output image in ARGB8888 or RGB8888 mode  

**GREEN[7:0]**  
Bits 15:8  **GREEN[7:0]**: Green value of the output image in ARGB8888 or RGB8888 mode  

**BLUE[7:0]**  
Bits 7:0  **BLUE[7:0]**: Blue value of the output image in ARGB8888 or RGB8888 mode
15.5.16 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x038
Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

RGB565 color mode

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

RED[4:0] | GREEN[5:0] | BLUE[4:0]

rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 RED[4:0]: Red value of the output image in RGB565 mode

Bits 10:5 GREEN[5:0]: Green value of the output image in RGB565 mode

Bits 4:0 BLUE[4:0]: Blue value of the output image in RGB565 mode

15.5.17 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x038
Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB1555 color mode

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

A | RED[4:0] | GREEN[4:0] | BLUE[4:0]

rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 A: Alpha channel value of the output color in ARGB1555 mode

Bits 14:10 RED[4:0]: Red value of the output image in ARGB1555 mode

Bits 9:5 GREEN[4:0]: Green value of the output image in ARGB1555 mode

Bits 4:0 BLUE[4:0]: Blue value of the output image in ARGB1555 mode
15.5.18 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x038
Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB4444 color mode

15.5.19 DMA2D output memory address register (DMA2D_OMAR)

Address offset: 0x03C
Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.
15.5.20 DMA2D output offset register (DMA2D_OOR)

Address offset: 0x040
Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

LO[15:0]

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 LO[15:0]: Line offset

This field gives the line offset used for the output, expressed:

- in pixels when LOM = 0 in DMA2D_CR. Only LO[13:0] bits are considered, LO[15:14] bits are ignored.
- in bytes when LOM = 1

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

15.5.21 DMA2D number of line register (DMA2D_NLR)

Address offset: 0x044
Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

PL[13:0]

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 PL[13:0]: Pixel per lines per lines of the area to be transferred
If any of the input image format is 4-bit per pixel, pixel per lines must be even.

Bits 15:0 NL[15:0]: Number of lines of the area to be transferred.
### 15.5.22 DMA2D line watermark register (DMA2D_LWR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**LW[15:0]**

rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LW[15:0]**: Line watermark for interrupt generation

An interrupt is raised when the last pixel of the watermarked line has been transferred.

### 15.5.23 DMA2D AXI master timer configuration register (DMA2D_AMTCR)

Address offset: 0x04C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DT[7:0]**

rw rw rw rw rw rw rw rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DT[7:0]**: Dead time

Dead time value in the AXI clock cycle inserted between two consecutive accesses on the AXI master port. These bits represent the minimum guaranteed number of cycles between two consecutive AXI accesses.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Dead-time functionality enable
### 15.5.24 DMA2D foreground CLUT (DMA2D_FGCLUTx)

Address offset: $0x400 + 0x4 \times x$, ($x = 0$ to 255)

Reset value: $0xXXXX XXXX$

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 **ALPHA[7:0]**: Alpha  
Alpha value for index $\{x\}$ for the foreground

Bits 23:16 **RED[7:0]**: Red  
Red value for index $\{x\}$ for the foreground

Bits 15:8 **GREEN[7:0]**: Green  
Green value for index $\{x\}$ for the foreground

Bits 7:0 **BLUE[7:0]**: Blue  
Blue value for index $\{x\}$ for the foreground

### 15.5.25 DMA2D background CLUT (DMA2D_BGCLUTx)

Address offset: $0x800 + 0x4 \times x$, ($x = 0$ to 255)

Reset value: $0xXXXX XXXX$

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 **ALPHA[7:0]**: Alpha  
Alpha value for index $\{x\}$ for the background

Bits 23:16 **RED[7:0]**: Red  
Red value for index $\{x\}$ for the background

Bits 15:8 **GREEN[7:0]**: Green  
Green value for index $\{x\}$ for the background

Bits 7:0 **BLUE[7:0]**: Blue  
Blue value for index $\{x\}$ for the background
### 15.5.26 DMA2D register map

#### Table 128. DMA2D register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DMA2D_CR</td>
<td>MODE[2:0] CEE CFGE CAEIE CAFIE TWE TEIE LOM ABORT SUSP START</td>
</tr>
</tbody>
</table>
| 0x004  | DMA2D_ISR             | CEIF CTFIF CTIFIF CTFI TCWIF CTFIF CTIF IF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIF CIA...
Refer to Section 2.3 for the register boundary addresses.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x038</td>
<td>DMA2D_GCLR(ARGB4444 color mode)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x03C</td>
<td>DMA2D_OMAR(MA[31:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x040</td>
<td>DMA2D_OOR(LO[15:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x044</td>
<td>DMA2D_NLR(PL[13:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x048</td>
<td>DMA2D_LWR(LW[15:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x04C</td>
<td>DMA2D_AMTCR(DT[7:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x055-0x3FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x400</td>
<td>DMA2D_FGCLUTx(ALPHA[7:0], RED[7:0], GREEN[7:0], BLUE[7:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>0x800</td>
<td>DMA2D_BGCLUTx(ALPHA[7:0], RED[7:0], GREEN[7:0], BLUE[7:0])</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
16 Neo-Chrom graphic processor (GPU2D)

16.1 GPU2D introduction

The GPU2D is a dedicated graphics processing unit accelerating numerous 2.5D graphics applications such as graphical user interface (GUI), menu display or animations.

The GPU2D works together with an optimized software stack designed for state of the art graphic rendering.

16.2 GPU2D main features

Main features
- Multi-threaded fragment (pixel) processing core with a VLIW (very-long instruction word) instruction set
- Fixed point functional units
- Command list based DMAs to minimize CPU overhead
- Two 64-bit AXI master interfaces for texture and framebuffer access
- Dedicated 64-bit AXI master interface for command list
- 32-bit AHB slave interface for register bank access
- Up to 4 general-purpose flags for system-level synchronization
- Texture decompression unit with TSC™4 and TSC™6/TSC™6a support

2D drawing features
- Pixel/line drawing
- Filled rectangles
- Triangles, quadrilateral drawing
- Anti-aliasing 8xMSAA (multi-sample anti-aliasing)

Image transformations
- 3D perspective correct projections
- Texture mapping with bilinear filtering or point sampling

Blit support
- Rotation, mirroring, stretching (independently on x and y axis)
- Source and/or destination color keying
- Pixel format conversions

Text rendering support
- A1, A2, A4, and A8 bitmap anti-aliased
- Subsampled anti-aliased
Color formats
- ABGR8888, ARGB8888, BGRA8888, RGBA8888
- xBGR8888, xRGB8888, BGRx8888, RGBx8888, RGB888, BGR888
- BGR565, RGB565
- RGB322, BGR322
- TSC4, TSC6, TSC6A
- L1, L2, L4, L8 (grayscale)
- A1, A2, A4, A8

Full alpha blending with hardware blender
- Programmable blending modes
- Source/destination color keying

16.3 GPU2D general description

16.3.1 GPU2D block diagram

![Figure 137. GPU2D block diagram](MSv66000V1)

16.3.2 GPU2D pins and internal signals

The internal signals of the GPU2D are given in the table below.

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpu2d_aclk</td>
<td>Input</td>
<td>GPU2D AXI clock</td>
</tr>
<tr>
<td>gpu2d_hclk</td>
<td>Input</td>
<td>GPU2D AHB clock</td>
</tr>
<tr>
<td>gpu2d_irq</td>
<td>Output</td>
<td>GPU2D interrupt request</td>
</tr>
</tbody>
</table>
Table 129. GPU2D internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpu2d_irqsys</td>
<td>Output</td>
<td>GPU2D system interrupt request</td>
</tr>
<tr>
<td>gpu2d_flag[3:0]</td>
<td>Output</td>
<td>GPU2D general purpose flags</td>
</tr>
</tbody>
</table>

Table 130. GPU2D trigger connections

<table>
<thead>
<tr>
<th>Trigger name</th>
<th>Direction</th>
<th>Trigger source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>gpu2d_flag[0]</td>
<td>Output</td>
<td>hpdma_trigsel[13]</td>
</tr>
</tbody>
</table>
17 Texture cache (ICACHE)

17.1 ICACHE introduction

The texture cache (ICACHE) is introduced on the AXI read-only texture port of the GPU to improve performance when reading texture data from internal and external memories.

The texture cache is an assembly of ICACHE (a peripheral with AHB ports) and an AXI-to-AHB bus bridge plugged on ICACHE AHB slave port, providing an AXI interface on the texture cache slave port.

The following sections only describe the AHB ICACHE peripheral itself.

Some specific features like hit-under-miss, and critical-word-first refill policy, allow close to zero-wait-state performance in most use cases.

17.2 ICACHE main features

The main features of ICACHE are described below:

- **Bus interface**
  - One 64-bit AHB slave port, the read port (input from GPU texture read interface)
  - One 64-bit AHB master port: master1 port (output to texture bus of main AXI/AHB bus matrix)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)

- **Cache access**
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, w, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile

- **the Replacement and refill**
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls

- **Performance counters**
  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)

- **Error management**
  - Possibility to detect an unexpected cacheable write access, to flag an error and optionally to raise an interrupt

- **Maintenance operation**
  - Cache invalidate: full cache invalidation, fast command, noninterruptible
17.3 ICACHE implementation

The purpose of the texture cache is to cache GPU read accesses to texture data. As such, the ICACHE only manages cacheable read transactions and does not manage cacheable write transactions.

The noncacheable transactions (both read and write ones) bypass the ICACHE.

For the error management purpose, in case a write cacheable transaction is presented (this only happens in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

17.4 ICACHE functional description

The purpose of the texture cache is to cache GPU read accesses to texture data. As such, the ICACHE only manages cacheable read transactions and does not manage cacheable write transactions.

The noncacheable transactions (both read and write ones) bypass the ICACHE.

For the error management purpose, in case a write cacheable transaction is presented (this only happens in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

Table 131. ICACHE features

<table>
<thead>
<tr>
<th>Feature</th>
<th>ICACHE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of ways</td>
<td>4</td>
</tr>
<tr>
<td>Cache size</td>
<td>16 Kbytes</td>
</tr>
<tr>
<td>Cache line width</td>
<td>32 bytes</td>
</tr>
<tr>
<td>Number of regions to remap</td>
<td>0</td>
</tr>
<tr>
<td>Data size of AHB slave interface</td>
<td>64 bits</td>
</tr>
<tr>
<td>Data size of AHB fast master1 interface</td>
<td>64 bits</td>
</tr>
<tr>
<td>Data size of AHB slow master2 interface</td>
<td>0</td>
</tr>
</tbody>
</table>
17.4.1 ICACHE block diagram

Figure 138. ICACHE block diagram

17.4.2 ICACHE reset and clocks

The ICACHE is clocked on the texture AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (=0).

Note: When disabled, the ICACHE is bypassed: slave input requests are just forwarded to the master port.
17.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:
- address tags that indicate which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:
- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see Table 132 for B and W definitions):
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. Figure 139 shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 132. TAG memory dimensioning parameters for n-way set associative operating mode (default)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache size</td>
<td>S Kbytes = s bytes (s = 1024 x S)</td>
<td>8 Kbytes = 8192 bytes</td>
</tr>
<tr>
<td>Cache number of ways</td>
<td>n</td>
<td>2</td>
</tr>
<tr>
<td>Cache line size</td>
<td>L-byte = l-bit (l = 8 x L)</td>
<td>16-byte = 128-bit</td>
</tr>
<tr>
<td>Number of cache lines (per way)</td>
<td>LpW = s / (n x L) lines / way</td>
<td>256 lines / way</td>
</tr>
<tr>
<td>Address byte offset size</td>
<td>B = log2(L) bit</td>
<td>4-bit</td>
</tr>
<tr>
<td>Address way index size</td>
<td>W = log2(LpW) bit</td>
<td>8-bit</td>
</tr>
<tr>
<td>TAG address size</td>
<td>T = (32 - W - B) bit</td>
<td>20-bit</td>
</tr>
</tbody>
</table>
17.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cache size</td>
<td>$S$ Kbytes $= s$ bytes ($s = 1024 \times S$)</td>
<td>8 Kbytes = 8192 bytes</td>
</tr>
<tr>
<td>Cache number of ways</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Cache line size</td>
<td>$L$-byte $= l$-bit ($l = 8 \times L$)</td>
<td>16-byte $= 128$-bit</td>
</tr>
<tr>
<td>Number of cache lines</td>
<td>$LpW = s / L$ lines</td>
<td>512 lines</td>
</tr>
<tr>
<td>Address byte offset size</td>
<td>$B = \log_2(L)$ bit</td>
<td>4-bit</td>
</tr>
<tr>
<td>Address way index size</td>
<td>$W = \log_2(LpW)$ bit</td>
<td>9-bit</td>
</tr>
<tr>
<td>TAG address size</td>
<td>$T = (32 - W - B)$ bit</td>
<td>19-bit</td>
</tr>
</tbody>
</table>
All cache operations (such as read, refill, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in Section 17.4.7): only one way (the unique one) is possible for any data refill.

### 17.4.5 ICACHE enable

To activate the ICACHE, the EN bit must be set to 1 in ICACHE_CR.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave port to the master port in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This invalidate procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The ICACHE is by default disabled at boot.

### 17.4.6 Cacheable and noncacheable traffic

The ICACHE is placed on the GPU texture bus, and thus caches all internal and external memory regions (ranging from address 0x0000 0000 to 0x3FFF FFFF, respectively 0x6000 0000 to 0x9FFF FFFF, of the memory map).

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in Table 134. This AHB attribute depends on the GPU setting for the addressed region.

**Table 134. ICACHE cacheability for AHB transaction**

<table>
<thead>
<tr>
<th>AHB lookup attribute</th>
<th>Cacheability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cacheable</td>
</tr>
<tr>
<td>0</td>
<td>Noncacheable</td>
</tr>
</tbody>
</table>

In the case of a noncacheable access (either a noncacheable read or a noncacheable write), the ICACHE is bypassed, meaning that the AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in Section 17.4.7.
17.4.7 Cacheable accesses

When the ICACHE receives a cacheable transaction from the GPU, the ICACHE checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.
  - The critical-word-first policy insures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).
  - The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words).
  - The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port.
- The address is not present in TAG memory, but belongs to the refill burst from the main memory that is currently ongoing: **cache hit** (hit-under-miss feature).
  - This happens during cache-line refill. The ICACHE can provide the requested data as soon as the data is available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If ever the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

17.4.8 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.
17.4.9 **ICACHE performance monitoring**

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.
  
  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see Section 17.4.7).

- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.
  
  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

17.4.10 **ICACHE boot**

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

17.5 **ICACHE low-power modes**

At device level, using the ICACHE reduces the power consumption by reading textures from the internal ICACHE most of the time, rather than from the bigger and then more-power-consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with a lower-performance profile (in terms of hit ratio) and stringent low-power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see Figure 139). The power consumption is then reduced by accessing, for each request, only the necessary cut of TAG and data memories. Meanwhile, the cache effect still improves memory access performance. Even if for most texture accesses, it is a little less efficient than with an n-way set associative cache mode.
17.6 ICACHE error management and interrupts

In case an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is then generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICACHE</td>
<td>Functional error</td>
<td>ERRF in ICACHE_SR</td>
<td>ERRIE in ICACHE_IER</td>
<td>Set CERRF to 1 in ICACHE_FCR</td>
</tr>
<tr>
<td></td>
<td>End of busy state (invalidate finished)</td>
<td>BSYENDF in ICACHE_SR</td>
<td>BSYENDIE in ICACHE_IER</td>
<td>Set CBSYENDF to 1 in ICACHE_FCR</td>
</tr>
</tbody>
</table>

The ICACHE also propagates all AHB bus errors (such as address decoding issues) from the master1 port back to the slave read port.

17.7 ICACHE registers

17.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor
Bit 17 **MISSMEN**: miss monitor enable  
0: cache miss monitor switched off. Stopping the monitor does not reset it.  
1: cache miss monitor enabled  

Bit 16 **HITMEN**: hit monitor enable  
0: cache hit monitor switched off. Stopping the monitor does not reset it.  
1: cache hit monitor enabled  

Bits 15:3 Reserved, must be kept at reset value.  

Bit 2 **WAYSEL**: cache associativity mode selection  
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).  
0: direct mapped cache (1-way cache)  
1: n-way set associative cache (reset value)  

Bit 1 **CACHEINV**: cache invalidation  
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.  
0: no effect  
1: invalidate entire cache (all cache lines valid bit = 0)  

Bit 0 **EN**: enable  
0: cache disabled  
1: cache enabled  

### 17.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 2 **ERRF**: cache error flag  
0: no error  
1: an error occurred during the operation (cacheable write)  

Bit 1 **BSYENDF**: busy end flag  
0: cache busy  
1: full invalidate CACHEINV operation finished  

Bit 0 **BUSYF**: busy flag  
0: cache not busy on a CACHEINV operation  
1: cache executing a full invalidate CACHEINV operation
### 17.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 2 ERRIE**: interrupt enable on cache error  
  Set by software to enable an interrupt generation in case of cache functional error (cacheable write access).  
  - 0: interrupt disabled on error  
  - 1: interrupt enabled on error

- **Bit 1 BSYENDIE**: interrupt enable on busy end  
  Set by software to enable an interrupt generation at the end of a cache invalidate operation.  
  - 0: interrupt disabled on busy end  
  - 1: interrupt enabled on busy end

- **Bit 0** Reserved, must be kept at reset value.

### 17.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 2 CERRF**: clear cache error flag  
  Set by software.  
  - 0: no effect  
  - 1: clears ERRF flag in ICACHE_SR

- **Bit 1 CBSYENDF**: clear busy end flag  
  Set by software.  
  - 0: no effect  
  - 1: clears BSYENDF flag in ICACHE_SR.
Bit 0  Reserved, must be kept at reset value.

### 17.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010
Reset value: 0x0000 0000

![hit_monitor_register](image)

Bits 31:0  **HITMON[31:0]**: cache hit monitor counter

### 17.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014
Reset value: 0x0000 0000

![miss_monitor_register](image)

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **MISSMON[15:0]**: cache miss monitor counter

### 17.7.7 ICACHE register map

#### Table 136. ICACHE register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>ICACHE_CR</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>ICACHE_SR</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x008</td>
<td>ICACHE_IER</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

822/3791 RM0477 Rev 6
Refer to Section 2.3 for the register boundary addresses.
18 Graphic timer (GFXTIM)

18.1 Introduction

The graphic timer (GFXTIM) is a graphic oriented timer allowing smart management of graphical events for frame or line counting.

18.2 GFXTIM main features

- Integrated frame and line clock generation
- One absolute frame counter with one compare channel
- Two auto-reload relative frame counters
- One line timer with two compare channels
- External tearing-effect line management and synchronization
- Four programmable event generators with external trigger generation
- One watchdog counter

18.3 GFXTIM functional description

18.3.1 Block diagram

The graphic timer is split into six functional blocks
- Clock generator
- Absolute timers
- Relative timers
- Tearing-effect detection
- Event generators
- Watchdog timer
**18.3.2 GFXTIM pins and internal signals**

Table 137. GFXTIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFXTIM_TE</td>
<td>Input</td>
<td>Tearing effect</td>
</tr>
<tr>
<td>GFXTIM_FCKCAL</td>
<td>Output</td>
<td>Frame clock calibration output</td>
</tr>
<tr>
<td>GFXTIM_LCKCAL</td>
<td>Output</td>
<td>Line clock calibration output</td>
</tr>
</tbody>
</table>

Table 138. GFXTIM internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gfxtim_hclk</td>
<td>Digital input</td>
<td>Kernel and register interface clock</td>
</tr>
<tr>
<td>gfxtim_it</td>
<td>Digital output</td>
<td>Global interrupt</td>
</tr>
<tr>
<td>gfxtim_wit</td>
<td>Digital output</td>
<td>Watchdog global interrupt</td>
</tr>
<tr>
<td>gfxtim_ev1</td>
<td>Digital output</td>
<td>Graphic timer event 1</td>
</tr>
<tr>
<td>gfxtim_ev2</td>
<td>Digital output</td>
<td>Graphic timer event 2</td>
</tr>
<tr>
<td>gfxtim_ev3</td>
<td>Digital output</td>
<td>Graphic timer event 3</td>
</tr>
<tr>
<td>gfxtim_ev4</td>
<td>Digital output</td>
<td>Graphic timer event 4</td>
</tr>
<tr>
<td>gfxtim_wrld</td>
<td>Digital input</td>
<td>Watchdog reload</td>
</tr>
<tr>
<td>gfxtim_ite</td>
<td>Digital input</td>
<td>Internal tearing effect</td>
</tr>
</tbody>
</table>
The table below shows how GFXTIM triggers are connected.

### Table 139. GFXTIM trigger interconnections

<table>
<thead>
<tr>
<th>Trigger name</th>
<th>Direction</th>
<th>Trigger source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>gfxtim_ev1</td>
<td>Output</td>
<td>hpdma_trigsel[20]</td>
</tr>
<tr>
<td>gfxtim_ev2</td>
<td>Output</td>
<td>hpdma_trigsel[19]</td>
</tr>
<tr>
<td>gfxtim_ev3</td>
<td>Output</td>
<td>hpdma_trigsel[18]</td>
</tr>
<tr>
<td>gfxtim_ev4</td>
<td>Output</td>
<td>hpdma_trigsel[17]</td>
</tr>
<tr>
<td>gfxtim_wrld</td>
<td>Input</td>
<td>reserved</td>
</tr>
<tr>
<td>gfxtim_ite</td>
<td>Input</td>
<td>reserved</td>
</tr>
<tr>
<td>gfxtim_hsync[0]</td>
<td>Input</td>
<td>LCD_HSYNC</td>
</tr>
<tr>
<td>gfxtim_hsync[1]</td>
<td>Input</td>
<td>dcmipp_evt_hsync[0]</td>
</tr>
<tr>
<td>gfxtim_hsync[2]</td>
<td>Input</td>
<td>reserved</td>
</tr>
<tr>
<td>gfxtim_hsync[3]</td>
<td>Input</td>
<td>reserved</td>
</tr>
<tr>
<td>gfxtim_vsync[0]</td>
<td>Input</td>
<td>LCD_VSYNC</td>
</tr>
<tr>
<td>gfxtim_vsync[1]</td>
<td>Input</td>
<td>dcmipp_evt_vsync[0]</td>
</tr>
<tr>
<td>gfxtim_vsync[2]</td>
<td>Input</td>
<td>reserved</td>
</tr>
<tr>
<td>gfxtim_vsync[3]</td>
<td>Input</td>
<td>reserved</td>
</tr>
</tbody>
</table>

### 18.3.3 Clock generator

Two clocks are generated internally to fed the absolute and relative timers:
- the frame clock: clocking frame counters
- the line clock: clocking line counters

#### Internal counter for time base generation

The GFXTIM embeds a two-clock generation counter:
- a 22-bit auto-reload down-counter on the system clock
- a 12-bit auto-reload down-counter on selectable internal or external event
Figure 141. Clock generator

Line clock counter

The line clock counter is a 22-bit auto-reload down-counter on the system clock.

The line clock counter is enabled selecting its clock source with LCCCS (line clock counter clock source) in GFXTIM_CGCR (clock generator control register).

The line clock counter can be reloaded automatically when one of the following event occurs:

- frame clock counter underflows
- VSYNC edge (with control of polarity)
- HSYNC edge (with control of polarity)
- TE edge (with control of polarity)

The hardware reload function and source can be selected with LCCHRS (line clock counter hardware reload source) in GFXTIM_CGCR (clock generator control register).

The line clock counter can be reloaded by software, setting LCCFR (line clock counter force reload) in GFXTIM_CGCR (clock generator control register).

The reload value is programmed in GFXTIM_LCCRR (line clock counter reload register).

Frame clock counter

The frame clock counter if an 12-bit auto-reload down-counter clocked by either, TE, HYSNC, VSYNC or a line clock counter underflow.

The frame clock counter is enabled selecting its clock source with FCCCS in GFXTIM_CGCR.

The frame clock counter can be reloaded automatically when one of the following event occurs:

- line clock counter underflow
- VSYNC edge (with control of polarity)
- HSYNC edge (with control of polarity)
- TE edge (with control of polarity)

The hardware reload function and source can be selected with FCCHRS in GFXTIM_CGCR.

The frame clock counter can be reloaded by software, setting FCCFR in GFXTIM_CGCR.
The reload value is programmed in GFXTIM_FCCRR.

**Clock generation**

The line clock source can be one of the following:
- underflow flag of the internal 22-bit down-counter
- underflow flag of the internal 12-bit down-counter
- HSYNC, VSYNC or TE pin (with control of polarity)

The frame clock source can be one of the following:
- underflow flag of the internal 22-bit down-counter
- underflow flag of the internal 12-bit down-counter
- HSYNC, VSYNC or TE pin (with control of polarity)

**Clock calibration output**

For calibration/debug purpose, the frame clock and line clock can be output on a specific I/O.

The frame clock calibration output is enabled by setting FCCOE (frame clock calibration output enable) in GFXTIM_CR (configuration register).

The line clock calibration output is enabled by setting LCCOE in GFXTIM_CR.

**Synchronization and tearing-effect sources**

The GFXTIM can be connected to peripherals providing HSYNC and/or VSYNC synchronization signals, like the LCD-TFT controller or the camera interface.

The source of HSYNC and/or VSYNC is selected through SYNCS (synchronization source) in GFXTIM_CR.

The tearing-effect source can be an external pin or can be provided by the MIPI® DSI Host on system embedding this interface. To extend the orthogonality versus the synchronization signals, the tearing-effect source can be also the selected HSYNC or VSYNC input.

The tearing-effect source can be selected with TES (TE source) in GFXTIM_CR.

### 18.3.4 Example of clock generator configuration

The clock generator can have several configuration to work:
- in standalone (without any external synchronization)
- with external HSYNC and VSYNC
- with external HSYNC only
- with external VSYNC only
- with external CSYNC (TE) only

The synchronization signals, HSYNC, VSYNC and TE, are symmetrical in the implementation and can be exchanged if needed.

The set of examples detailed below are given for reference, but other combinations can be programmed into the clock generator.
Standalone

In standalone configuration, the clock generator provides to the GFXTIM, the frame clock and the line clock without any external signals.

Figure 142. Waveforms in standalone

The LCCUF (line-clock counter underflow) event acts as line clock and the FCCUF (frame-clock counter underflow) acts as frame clock.

The clock generator can be synchronized by one of the following ways:
- by software reload, setting the FCCFR in GFXTIM_CGCR.
- by hardware reload on an external VSYNC (or TE)

Figure 143. Active counters and signals in standalone

External HSYNC and VSYNC

When using external HSYNC and VSYNC, the counters are not used for the line and frame clock generation. The clock generator copy directly HSYNC to the line clock and VSYNC to the frame clock.

Figure 144. Waveforms with external HSYNC and VSYNC
HSYNC acts as line clock and VSYNC acts as frame clock.

**External HSYNC only**

With external HSYNC only, the clock generator provides to the GFXTIM, the frame clock and the line clock based only on HSYNC.

**Figure 145. Waveforms with external HSYNC only**

![Waveforms with external HSYNC only](image1)

HSYNC acts as line clock and FCCUF acts as frame clock.

The clock generator can be synchronized by one of the following ways:
- by software reload, setting FCCFR in GFXTIM_CGCR
- by hardware reload on an external VSYNC (or TE)

**Figure 146. Active counters and signals with external HSYNC only**

![Active counters and signals with external HSYNC only](image2)

**External VSYNC only**

With external VSYNC only, the clock generator provides to the GFXTIM, the frame clock and the line clock based only on VSYNC.

**Figure 147. Waveforms with external VSYNC only**

![Waveforms with external VSYNC only](image3)

LCCUF (line clock counter underflow) acts as line clock and VSYNC acts as frame clock.
The line clock counter generating the line clock is reloaded on VSYNC event.

**Figure 148. Active counters with external VSYNC only**

The line clock counter can also act as a prescaler for the frame clock counter to have a wider range of counting. FCCUF acts as line clock.

**Figure 149. Prescaling when external VSYNC only**

**External CSYNC only**

With external CSYNC only, the clock generator provides to the GFXTIM, the frame clock and the line clock based only on CSYNC (in the following figures, CSYNC is input on TE).

**Figure 150. Waveforms with external CSYNC only**

The CSYNC (on TE pin in this example) acts as line clock and LCCUF acts as frame clock. The line clock counter generating the frame clock is reloaded on CSYNC event.
The line-clock counter can also act as a prescaler for the frame clock counter to have a wider range of counting. FCCUF acts as frame clock.

18.3.5 Absolute timers

Absolute time

The absolute time 32-bit value can be read directly in GFXTIM_ATR (absolute time register) to provide a global time and to generate absolute time accurate event to synchronize the graphical software.

The absolute time is generated by the combination of:
- the absolute frame counter
- the absolute line counter

Absolute frame counter

The absolute frame counter is a 20-bit free running up-counter with a software controlled reset.

It can be enabled by writing 1 in AFCEN (absolute-frame counter enable) of GFXTIM_TCR (timers configuration register).
The counter is incremented at each rising edge of the frame clock and is reset by setting FAFCR (force absolute-frame counter reset) in GFXTIM_TCR.

In case of overflow of the absolute frame counter, AFCOF (overflow flag) is set in GFXTIM_ISR (interrupt status register), and an interrupt is generated if AFCOIE (overflow interrupt enable) is set in (interrupt enable register).

A 20-bit compare register can generate an absolute-frame counter compare event when the counter value matches the compare value. When a compare event occurs, AFCC1F (compare 1 flag) is set in GFXTIM_ISR, and an interrupt is generated if AFCC1IE (compare 1 interrupt enable) is set in GFXTIM_IER.

The absolute frame counter can be disabled by writing 1 in AFCDIS (counter disable) of GFXTIM_TDR (timers disable register).

The status (enabled/disabled) can be monitored through AFCS (counter status) in GFXTIM_TSR (timers status register).

### Absolute line counter

The absolute line counter is 12-bit free running-up counter with an hardware controlled reset.

It can be enabled by writing 1 in ALCEN of GFXTIM_TCR.

The counter is incremented at each rising edge of the line clock an is reset at each rising edge of the frame clock or setting FALCR in GFXTIM_TCR).

In case of overflow of the absolute line counter, ALCOF is set in GFXTIM_ISR, and an interrupt is generated if ALCOIE is set in GFXTIM_IER.

Two 12-bit compare registers can generate an absolute line counter compare events when the counter value matches the compare value. When a compare event occurs, ALCC1F or ALCC2F is set in GFXTIM_ISR, and an interrupt is generated if ALCC1IE or ALCC2IE is set in GFXTIM_IER.

The absolute line counter can be disabled writing 1 in ALCDIS of GFXTIM_TDR.

The status (enabled/disabled) can be monitored through ALCS in GFXTIM_TSR.

### 18.3.6 Relative timers

The relative timers can generate periodic events to synchronize the graphical tasks at frame level.

The relative frame counters are 12-bit down-counting auto-reload timers that are decremented at each rising edge of the frame clock. The counter is started by writing 1 in RFCxEN (relative frame counter x enable) of GFXTIM_TCR. The starting value is automatically loaded from GFXTIM_RFCxRR (relative frame counter x reload register).

The current relative frame counter value can be read directly in GFXTIM_RFCxR.

Once the counter reaches zero, it is automatically reloaded with the value of GFXTIM_RFCxRR. RFCxRF (reload flag) is set in GFXTIM_ISR is set, and an interrupt is generated if RFCxRIE (interrupt enable) is set in GFXTIM_IER.

Once the reload operation occurs, the timer can be disabled automatically or continue counting if RFCxCM (continuous mode) is set in GFXTIM_TCR.

The status (enabled/disabled) can be monitored through RFCxS in GFXTIM_TSR.
The counter can be reloaded on the fly by setting FRFCRx in GFXTIM_TCR. This force reload neither stop the timer (even if RFCxCM = 0 in GFXTIM_TCR), nor set RFCxRF (reload flag) in GFXTIM_ISR.

The relative frame counter can be disabled by setting RFCxDIS (disable) in GFXTIM_TDR.

18.3.7 **Tearing-effect detection**

A tearing-effect line can work in one of the two configurations shown in the figure below.

**Figure 153. Tearing-effect configurations**

A tearing-effect event can be generated on rising or falling edge depending on TEPOL (tearing-effect polarity) in GFXTIM_CR.

When a tearing-effect event is detected, TEF (tearing-event flag) is set in GFXTIM_ISR, and an interrupt is generated if TEIE (interrupt enable) is set in GFXTIM_IER.

18.3.8 **Event generator**

The event generator can combine timer events into complex events. Up to four combined events can be generated.

The events can be used for:
- interrupt generation
- watchdog clocking
- external trigger generation

A complex event is a combination between a frame event and a line event.

Once a frame event occurs, the GFXTIM waits for the line event to occur before generating the complex event.

The frame event is selected by the corresponding FESx (frame event selection x) in GFXTIM_EVSR (events selection y register). The line event is selected by the corresponding LESx in GFXTIM_EVSR.

The complex event generation is enabled by setting the corresponding EVxEN (event x enable) in GFXTIM_EVCR (event control register).

It is recommended to disable the event generation prior to any event configuration to avoid spurious complex event generation.

When a complex event occurs, the corresponding EVxF (event x flag) is set in GFXTIM_ISR, and an interrupt is generated if EVxIE (event x interrupt enable) is set in GFXTIM_IER.

Each of the events can be connected to another peripherals (such as DMA) to generate hardware triggers.
18.3.9 Watchdog timer

The watchdog timer is a 16-bit auto-reload down-counter with a programmable clock source.

**Clock source**

The watchdog clock source can be selected through WDGCS (clock source) in GTXTIM_WDGTCR (watchdog timer configuration register), between one of the following:
- line clock
- frame clock
- HSYNC
- VSYNC
- TE
- event 1
- event 2
- event 3
- event 4

**Startup**

The watchdog is started by setting WDGEN in GFXTIM_WDGTCR and stops automatically when reaching 0.

On start, the watchdog counter is automatically loaded with the auto-reload value programmed in GFXTIM_WDGRR (watchdog reload register).

The current watchdog value can be read through GFXTIM_WDGCR.

**Auto-reload**

The auto-reload can be forced in one of the following ways:
- by software, setting FWDGR (force watchdog reload) in GTXTIM_WDGTCR (watchdog timer configuration register)
- by hardware through an external trigger (gfxtim_wrld signal)

The polarity of the trigger is configured by WDGHRC (watchdog hardware reload configuration) in GFXTIM_WDGTCR.
Event and interrupt generation

The watchdog can generate two events:
- an alarm when the watchdog down-counter reaches 0
  The watchdog counter is automatically stopped, and WDGAF (alarm flag) is set in GFXTIM_ISR. An interrupt is generated if WDGAIE (alarm interrupt enable) is set in GFXTIM_IER.
- a pre-alarm when the watchdog counter matches the pre-alarm register value in GFXTIM_WDGPAR (watchdog pre-alarm register)
  WDGPF (pre-alarm flag) is set in GFXTIM_ISR, and an interrupt is generated if WDGPIE (pre-alarm interrupt enable) is set in GFXTIM_IER.

18.4 GFXTIM interrupts

An interrupt can be produced on the following events:
- absolute frame counter compare event
- absolute frame counter overflow
- absolute line counter compare events
- absolute line counter overflow
- relative frame counter reload events
- external tearing effect
- combined events
- watchdog alarm event
- watchdog pre-alarm event

Separate interrupt enable bits are available for flexibility.

Table 140. Graphic timer interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFXTIM</td>
<td>Absolute frame counter overflow</td>
<td>AFCOF</td>
<td>AFCOIE</td>
<td>write 1 in CAFCOF</td>
</tr>
<tr>
<td></td>
<td>Absolute frame counter compare 1</td>
<td>AFCC1F</td>
<td>AFCC1IE</td>
<td>write 1 in CAFCC1F</td>
</tr>
<tr>
<td></td>
<td>Absolute line counter overflow</td>
<td>ALCOF</td>
<td>ALCOIE</td>
<td>write 1 in CALCOF</td>
</tr>
<tr>
<td></td>
<td>Absolute line counter compare 1</td>
<td>ALCC1F</td>
<td>ALCC1IE</td>
<td>write 1 in CALCC1F</td>
</tr>
<tr>
<td></td>
<td>Absolute line counter compare 2</td>
<td>ALCC2F</td>
<td>ALCC2IE</td>
<td>write 1 in CALCC2F</td>
</tr>
<tr>
<td></td>
<td>Relative frame counter 1 reload</td>
<td>RFC1RF</td>
<td>RFC1RIE</td>
<td>write 1 in CRFC1RF</td>
</tr>
<tr>
<td></td>
<td>Relative frame counter 2 reload</td>
<td>RFC2RF</td>
<td>RFC2RIE</td>
<td>write 1 in CRFC2RF</td>
</tr>
<tr>
<td></td>
<td>External tearing effect</td>
<td>TEF</td>
<td>TEIE</td>
<td>write 1 in CTEF</td>
</tr>
<tr>
<td></td>
<td>Event 1</td>
<td>EV1F</td>
<td>EV1IE</td>
<td>write 1 in CEV1F</td>
</tr>
<tr>
<td></td>
<td>Event 2</td>
<td>EV2F</td>
<td>EV2IE</td>
<td>write 1 in CEV2F</td>
</tr>
<tr>
<td></td>
<td>Event 3</td>
<td>EV3F</td>
<td>EV3IE</td>
<td>write 1 in CEV3F</td>
</tr>
<tr>
<td></td>
<td>Event 4</td>
<td>EV4F</td>
<td>EV4IE</td>
<td>write 1 in CEV4F</td>
</tr>
</tbody>
</table>
Table 140. Graphic timer interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>GFXTIMW</td>
<td>Watchdog alarm</td>
<td>WDGAF</td>
<td>WDGAIE</td>
<td>write 1 in CWDGAF</td>
</tr>
<tr>
<td></td>
<td>Watchdog pre-alarm</td>
<td>WDGPF</td>
<td>WDGPIE</td>
<td>write 1 in CWDGPF</td>
</tr>
</tbody>
</table>

18.5  **GFXTIM registers**

18.5.1  **GFXTIM configuration register (GFXTIM_CR)**

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>LCCOE</th>
<th>FCCOE</th>
<th>SYNCS[1:0]</th>
<th>TEPOL</th>
<th>TES[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
</tr>
<tr>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 17 **LCCOE**: Line-clock calibration output enable
This bit enables the line-clock output.
0: Line-clock output disabled
1: Line-clock output enabled

Bit 16 **FCCOE**: Frame-clock calibration output enable
This bit enables the frame-clock output.
0: Frame-clock output disabled
1: Frame-clock output enabled

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:8 **SYNCS[1:0]**: Synchronization source
This bitfield selects the synchronization signals (HSYNC and VSYNC) sources.
00: gftim_hsync[0] and gftim_vsync[0] selected
01: gftim_hsync[1] and gftim_vsync[1] selected

Bits 7:5  Reserved, must be kept at reset value.

Bit 4 **TEPOL**: Tearing--effect polarity
This bit selects the tearing-effect polarity.
0: Tearing effect active on rising edge
1: Tearing effect active on falling edge

Bits 3:2  Reserved, must be kept at reset value.
### 18.5.2 GFXTIM clock generator configuration register (GFXTIM_CGCR)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>rw</td>
<td>Res</td>
<td>rw</td>
<td>Res</td>
<td>rw</td>
<td>Res</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Bit 31 Reserved, must be kept at reset value.  
Bits 30:28 **FCCHR[2:0]**: Frame clock counter hardware reload source  
This bitfield configures the hardware reload source for the frame clock counter.  
000: No hardware reload  
001: Line- -clock counter underflow  
010: HSYNC rising edge  
011: HSYNC falling edge  
100: VSYNC rising edge  
101: VSYNC falling edge  
110: TE rising edge  
111: TE falling edge  

Bits 27:25 Reserved, must be kept at reset value.  
Bit 24 **FCCFR**: Frame clock counter force reload  
This bit forces frame clock counter reload  
0: No effect  
1: Frame clock counter reload forced  

Bit 23 Reserved, must be kept at reset value.  
Bits 22:20 **FCCCS[2:0]**: Frame clock counter clock source  
This bitfield configures the clock source for the frame clock counter.  
000: Frame clock counter disabled  
001: Line clock counter underflow  
010: HSYNC rising edge  
011: HSYNC falling edge  
100: VSYNC rising edge  
101: VSYNC falling edge  
110: TE rising edge  
111: TE falling edge  

Bit 19 Reserved, must be kept at reset value.
Bits 18:16 **FCS[2:0]**: Frame clock source
This bitfield configures the frame clock source.
- 000: Line clock counter underflow
- 001: Frame clock counter underflow
- 010: HSYNC rising edge
- 011: HSYNC falling edge
- 100: VSYNC rising edge
- 101: VSYNC falling edge
- 110: TE rising edge
- 111: TE falling edge

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **LCCHR[2:0]**: Line clock counter hardware reload source
This bitfield configures the hardware reload source for the line clock counter.
- 000: No hardware reload
- 001: Frame clock counter underflow
- 010: HSYNC rising edge
- 011: HSYNC falling edge
- 100: VSYNC rising edge
- 101: VSYNC falling edge
- 110: TE rising edge
- 111: TE falling edge

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **LCCFR**: Line clock counter force reload
This bit forces line clock counter reload.
- 0: No effect
- 1: Line clock counter reload forced

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **LCCCS**: Line clock counter clock source
This bit configures the clock source for the line clock counter.
- 0: Line clock counter disabled
- 1: System clock selected

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LCS[2:0]**: Line clock source
This bitfield configures the line clock source.
- 000: Line clock counter underflow
- 001: Frame clock counter underflow
- 010: HSYNC rising edge
- 011: HSYNC falling edge
- 100: VSYNC rising edge
- 101: VSYNC falling edge
- 110: TE rising edge
- 111: TE falling edge
### 18.5.3 GFXTIM timers configuration register (GFXTIM_TCR)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>rw</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

- **Bits 31:23** Reserved, must be kept at reset value.
- **Bit 22** **FRFC2R**: Force relative frame counter 2 reload  
  This bit forces the reload of the relative frame counter 2.  
  0: No effect  
  1: Relative frame counter 2 reload forced
- **Bit 21** **RFC2CM**: Relative frame counter 2 continuous mode  
  This bit enables the continuous mode of the relative frame counter 2.  
  0: Relative frame counter 2 is one shot.  
  1: Relative frame counter 2 is in continuous mode.
- **Bit 20** **RFC2EN**: Relative frame counter 2 enable  
  This bit enables the relative frame counter 2.  
  0: No effect  
  1: Relative frame counter 2 enabled
- **Bit 19** Reserved, must be kept at reset value.
- **Bit 18** **FRFC1R**: Force relative frame counter 1 reload  
  This bit forces the reload of the relative frame counter 1.  
  0: No effect  
  1: Relative frame counter 1 reload forced
- **Bit 17** **RFC1CM**: Relative frame counter 1 continuous mode  
  This bit enables the continuous mode of the relative frame counter 1.  
  0: Relative frame counter 1 is one shot.  
  1: Relative frame counter 1 is in continuous mode.
- **Bit 16** **RFC1EN**: Relative frame counter 1 enable  
  This bit enables the relative frame counter 1.  
  0: No effect  
  1: Relative frame counter enabled
- **Bits 15:6** Reserved, must be kept at reset value.
- **Bit 5** **FALCR**: Force absolute line counter reset  
  This bit forces the reset of the absolute line counter.  
  0: No effect  
  1: Absolute line counter reset forced
Bit 4 **ALCEN**: Absolute line counter enable
This bit enables the absolute line counter.
0: No effect
1: Absolute line counter enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **FAFCR**: Force absolute frame counter reset
This bit forces the reset of the absolute frame counter.
0: No effect
1: Absolute frame counter reset forced

Bit 0 **AFCEN**: Absolute frame counter enable
This bit enables the absolute frame counter.
0: No effect
1: Absolute frame counter enabled

**18.5.4 GFXTIM timers disable register (GFXTIM_TDR)**

Address offset: 0x00C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **RFC2DIS**: Relative frame counter 2 disable
This bit disables the relative frame counter 2.
0: No effect
1: Relative frame counter 2 disabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RFC1DIS**: Relative frame counter 1 disable
This bit disables the relative frame counter 1.
0: No effect
1: Relative frame counter 1 disabled

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **ALCDIS**: Absolute line counter disable
This bit disables the absolute line counter.
0: No effect
1: Absolute line counter disabled

Bits 3:1 Reserved, must be kept at reset value.
18.5.5 **GFXTIM events control register (GFXTIM_EVCR)**

Address offset: 0x010

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:4</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 31</td>
<td>EV4EN: Event 4 enable</td>
</tr>
<tr>
<td></td>
<td>This bit enables the complex event 4 generation.</td>
</tr>
<tr>
<td></td>
<td>0: Event 4 generation disabled</td>
</tr>
<tr>
<td></td>
<td>1: Event 4 generation enabled</td>
</tr>
<tr>
<td>Bit 30</td>
<td>EV3EN: Event 3 enable</td>
</tr>
<tr>
<td></td>
<td>This bit enables the complex event 3 generation.</td>
</tr>
<tr>
<td></td>
<td>0: Event 3 generation disabled</td>
</tr>
<tr>
<td></td>
<td>1: Event 3 generation enabled</td>
</tr>
<tr>
<td>Bit 29</td>
<td>EV2EN: Event 2 enable</td>
</tr>
<tr>
<td></td>
<td>This bit enables the complex event 2 generation.</td>
</tr>
<tr>
<td></td>
<td>0: Event 2 generation disabled</td>
</tr>
<tr>
<td></td>
<td>1: Event 2 generation enabled</td>
</tr>
<tr>
<td>Bit 28</td>
<td>EV1EN: Event 1 enable</td>
</tr>
<tr>
<td></td>
<td>This bit enables the complex event 1 generation.</td>
</tr>
<tr>
<td></td>
<td>0: Event 1 generation disabled</td>
</tr>
<tr>
<td></td>
<td>1: Event 1 generation enabled</td>
</tr>
</tbody>
</table>

18.5.6 **GFXTIM events selection register (GFXTIM_EVSR)**

Address offset: 0x014

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:4</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 31</td>
<td>FES4[2:0]: Event 4 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 4 input source.</td>
</tr>
<tr>
<td>Bit 30</td>
<td>LES4[2:0]: Event 4 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 4 selection input source.</td>
</tr>
<tr>
<td>Bit 29</td>
<td>FES3[2:0]: Event 3 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 3 input source.</td>
</tr>
<tr>
<td>Bit 28</td>
<td>LES3[2:0]: Event 3 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 3 selection input source.</td>
</tr>
<tr>
<td>Bit 27</td>
<td>FES2[2:0]: Event 2 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 2 input source.</td>
</tr>
<tr>
<td>Bit 26</td>
<td>LES2[2:0]: Event 2 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 2 selection input source.</td>
</tr>
<tr>
<td>Bit 25</td>
<td>FES1[2:0]: Event 1 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 1 input source.</td>
</tr>
<tr>
<td>Bit 24</td>
<td>LES1[2:0]: Event 1 selection</td>
</tr>
<tr>
<td></td>
<td>This bit selects the event 1 selection input source.</td>
</tr>
</tbody>
</table>
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **FES4[2:0]:** Frame-event selection 4
This bitfield defines the frame-event selection for complex event 4 generation.
- 000: No frame event
- 001: Absolute frame counter overflow
- 010: Absolute frame counter compare
- 100: Relative frame counter 1 reload
- 101: Relative frame counter 2 reload
- Others: Reserved

Bit 27  Reserved, must be kept at reset value.

Bits 26:24  **LES4[2:0]:** Line-event selection 4
This bitfield defines the line-event selection for complex event 4 generation.
- 000: No line event
- 001: Absolute line counter overflow
- 010: Tearing effect
- 100: Absolute line counter 1 compare
- 101: Absolute line counter 2 compare
- Others: Reserved

Bit 23  Reserved, must be kept at reset value.

Bits 22:20  **FES3[2:0]:** Frame-event selection 3
This bitfield defines the frame-event selection for complex event 3 generation.
- 000: No frame event
- 001: Absolute frame counter overflow
- 010: Absolute frame counter compare
- 100: Relative frame counter 1 reload
- 101: Relative frame counter 2 reload
- Others: Reserved

Bit 19  Reserved, must be kept at reset value.

Bits 18:16  **LES3[2:0]:** Line-event selection 3
This bitfield defines the line-event selection for complex event 3 generation.
- 000: No line event
- 001: Absolute line counter overflow
- 010: Tearing effect
- 100: Absolute line counter 1 compare
- 101: Absolute line counter 2 compare
- Others: Reserved

Bit 15  Reserved, must be kept at reset value.

Bits 14:12  **FES2[2:0]:** Frame-event selection 2
This bitfield defines the frame-event selection for complex event 2 generation.
- 000: No frame event
- 001: Absolute frame counter overflow
- 010: Absolute frame counter compare
- 100: Relative frame counter 1 reload
- 101: Relative frame counter 2 reload
- Others: Reserved

Bit 11  Reserved, must be kept at reset value.
Bits 10:8 **LES2[2:0]:** Line-event selection 2
This bitfield defines the line-event selection for complex event 2 generation.
- 000: No line event
- 001: Absolute line counter overflow
- 010: Tearing effect
- 100: Absolute line counter 1 compare
- 101: Absolute line counter 2 compare
- Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **FES1[2:0]:** Frame-event selection 1
This bitfield defines the frame-event selection for complex event 1 generation.
- 000: No frame event
- 001: Absolute frame counter overflow
- 010: Absolute frame counter compare
- 100: Relative frame counter 1 reload
- 101: Relative frame counter 2 reload
- Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LES1[2:0]:** Line-event selection 1
This bitfield defines the line-event selection for complex event 1 generation.
- 000: No line event
- 001: Absolute line counter overflow
- 010: Tearing effect
- 100: Absolute line counter 1 compare
- 101: Absolute line counter 2 compare
- Others: Reserved

18.5.7 **GFXTIM watchdog timer configuration register**
*(GFXTIM_WDGTCR)*

Address offset: 0x020

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>FWDGR</td>
<td>Force watchdog reload</td>
</tr>
<tr>
<td>15:12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>11:9</td>
<td>WDGS</td>
<td>WDGE</td>
</tr>
<tr>
<td>8:6</td>
<td>WDGHC[1:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>5:3</td>
<td>WDGC[3:0]</td>
<td>Res.</td>
</tr>
<tr>
<td>2:0</td>
<td>Reserved</td>
<td>Res.</td>
</tr>
</tbody>
</table>

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

Bit 16 **FWDGR:** Force watchdog reload
This bit forces the reload of the graphic watchdog.
- 0: No effect
- 1: Graphic watchdog reload forced

Bits 15:12 Reserved, must be kept at reset value.
Bits 11:8  **WDGCS[3:0]**: Watchdog clock source
This bitfield selects the watchdog clock source.
0000: Line clock
0001: Frame clock
0010: HSYNC rising edge
0011: HSYNC falling edge
0100: VSYNC rising edge
0101: VSYNC falling edge
0110: TE rising edge
0111: TE falling edge
1000: Event 1
1001: Event 2
1010: Event 3
1011: Event 4
Others: Reserved

Bits 7:6  Reserved, must be kept at reset value.

Bits 5:4  **WDGHR[1:0]**: Watchdog hardware reload configuration
This bitfield configures the watchdog hardware reload.
00: Watchdog hardware reload disabled
01: Watchdog reloaded a rising edge of gfxtim_wrld
10: Watchdog reloaded a falling edge of gfxtim_wrld
11: Reserved

Bit 3  Reserved, must be kept at reset value.

Bit 2  **WDGS**: Watchdog status
This bit returns the status of the graphic watchdog.
0: Graphic watchdog disabled
1: Graphic watchdog enabled

Bit 1  **WDGDIS**: Watchdog disable
This bit disables the graphic watchdog.
0: No effect
1: Graphic watchdog disabled

Bit 0  **WDGEN**: Watchdog enable
This bit enables the graphic watchdog.
0: No effect
1: Graphic watchdog enabled
# GFXTIM interrupt status register (GFXTIM_ISR)

**Address offset:** 0x030  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:26</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

**Bit 25** WDGPF: Watchdog pre-alarm flag  
This bit indicates that a graphic watchdog pre-alarm occurred.  
0: No graphic watchdog pre-alarm occurred.  
1: A graphic watchdog pre-alarm occurred.

**Bit 24** WDGAF: Watchdog alarm flag  
This bit indicates that a graphic watchdog alarm occurred.  
0: No graphic watchdog alarm occurred.  
1: A graphic watchdog alarm occurred.

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

**Bit 19** EV4F: Event 4 flag  
This bit indicates a complex event 4 occurred.  
0: No complex event 4 occurred.  
1: A complex event 4 occurred.

**Bit 18** EV3F: Event 3 flag  
This bit indicates a complex event 3 occurred.  
0: No complex event 3 occurred.  
1: A complex event 3 occurred.

**Bit 17** EV2F: Event 2 flag  
This bit indicates a complex event 2 occurred.  
0: No complex event 2 occurred.  
1: A complex event 2 occurred.

**Bit 16** EV1F: Event 1 flag  
This bit indicates a complex event 1 occurred.  
0: No complex event 1 occurred.  
1: Complex event 1 occurred.

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

**Bit 13** RFC2RF: Relative frame counter 2 reload flag  
This bit indicates relative frame counter 2 has been reloaded.  
0: No reload occurred on relative frame counter 2.  
1: A reload on relative frame counter 2 occurred.
Bit 12 **RFC1RF**: Relative frame counter 1 reload flag
This bit indicates relative frame counter 1 has been reloaded.
0: No reload occurred on relative frame counter 1.
1: A reload on relative frame counter 1 occurred.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ALCC2F**: Absolute line counter compare 2 flag
This bit indicates match on compare 2 of the absolute line counter.
0: No match occurred on compare 2 of the absolute line counter.
1: A match on compare 2 of the absolute line counter occurred.

Bit 8 **ALCC1F**: Absolute line counter compare 1 flag
This bit indicates match on compare 1 of the absolute line counter.
0: No match occurred on compare 1 of the absolute line counter.
1: A match on compare 1 of the absolute line counter occurred.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **AFCC1F**: Absolute frame counter compare 1 flag
This bit indicates match on compare 1 of the absolute frame counter.
0: No match occurred on compare 1 of the absolute frame counter.
1: A match on compare 1 of the absolute frame counter occurred.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TEF**: Tearing-effect flag
This bit indicates a tearing effect event occurred.
0: No tearing effect occurred.
1: A tearing effect occurred.

Bit 1 **ALCOF**: Absolute line counter overflow flag
This bit indicates an overflow occurred on the absolute line counter.
0: No overflow occurred on the absolute line counter.
1: A overflow on the absolute line counter occurred.

Bit 0 **ACFOF**: absolutely frame counter overflow flag
This bit indicates an overflow occurred on the absolute frame counter.
0: No overflow occurred on the absolute frame counter.
1: An overflow on the absolute frame counter occurred.

### 18.5.9 **GFXTIM interrupt clear register (GFXTIM_ICR)**

Address offset: 0x034

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.
Bit 25  **CWDGPF**: Clear watchdog pre-alarm flag
        This bit clears WDGPF in GXFXTIM_ISR.
        0: No effect
        1: WDGPF cleared

Bit 24  **CWDGAF**: Clear watchdog alarm flag
        This bit clears WDGA in GXFXTIM_ISR.
        0: No effect
        1: WDGA cleared

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

Bit 19  **CEV4F**: Clear event 4 flag
        This bit clears EV4F in GXFXTIM_ISR.
        0: No effect
        1: EV4F cleared

Bit 18  **CEV3F**: Clear event 3 flag
        This bit clears EV3F in GXFXTIM_ISR.
        0: No effect
        1: EV3F cleared

Bit 17  **CEV2F**: Clear event 2 flag
        This bit clears EV2F in GXFXTIM_ISR.
        0: No effect
        1: EV2F cleared

Bit 16  **CEV1F**: Clear event 1 flag
        This bit clears EV1F in GXFXTIM_ISR.
        0: No effect
        1: EV1F cleared

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

Bit 13  **CRFC2RF**: Clear relative frame counter 2 reload flag
        This bit clears RFC2RF in GXFXTIM_ISR.
        0: No effect
        1: RFC2RF cleared

Bit 12  **CRFC1RF**: Clear relative frame counter 1 reload flag
        This bit clears RFC1RF in GXFXTIM_ISR.
        0: No effect
        1: RFC1RF cleared

Bits 11:10  **Reserved**, must be kept at reset value.

Bit 9  **CALCC2F**: Clear absolute line counter compare 2 flag
        This bit clears ALCC2F in GXFXTIM_ISR.
        0: No effect
        1: ALCC2F cleared

Bit 8  **CALCC1F**: Clear absolute line counter compare 1 flag
        This bit clears ALCC1F in GXFXTIM_ISR.
        0: No effect
        1: ALCC1F cleared

Bits 7:5  **Reserved**, must be kept at reset value.
Bit 4 **CAFCC1F**: Clear absolute frame counter compare 1 flag
This bit clears AFCC1F in GXTIM_ISR.
0: No effect
1: AFCC1F cleared

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CTEF**: Clear tearing-effect flag
This bit clears TEF in GXTIM_ISR.
0: No effect
1: TEF cleared

Bit 1 **CALCOF**: Clear absolute line counter overflow flag
This bit clears ALCOF in GXTIM_ISR.
0: No effect
1: ALCOF cleared

Bit 0 **CAFCOF**: Clear absolute frame counter overflow flag
This bit clears AFCOF in GXTIM_ISR.
0: No effect
1: AFCOF cleared

### 18.5.10 GFXTIM interrupt enable register (GFXTIM_IER)

Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 25 **WDGPIE**: Watchdog pre-alarm interrupt enable
This bit enables the watchdog pre-alarm interrupt generation.
0: Watchdog pre-alarm interrupt disabled
1: Watchdog pre-alarm interrupt enabled

Bit 24 **WDGAIE**: Watchdog alarm interrupt enable
This bit enables the watchdog alarm interrupt generation.
0: Watchdog alarm interrupt disabled
1: Watchdog alarm interrupt enabled

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

Bit 19 **EV4IE**: Event 4 interrupt enable
This bit enables the complex event 4 interrupt generation.
0: Event 4 interrupt disabled
1: Event 4 interrupt enabled
Bit 18  **EV3IE**: Event 3 interrupt enable  
This bit enables the complex event 3 interrupt generation.  
0: Event 3 interrupt disabled  
1: Event 3 interrupt enabled  

Bit 17  **EV2IE**: Event 2 interrupt enable  
This bit enables the complex event 2 interrupt generation.  
0: Event 2 interrupt disabled  
1: Event 2 interrupt enabled  

Bit 16  **EV1IE**: Event 1 interrupt enable  
This bit enables the complex event 1 interrupt generation.  
0: Event 1 interrupt disabled  
1: Event 1 interrupt enabled  

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

Bit 13  **RFC2RIE**: Relative frame counter 2 reload interrupt enable  
This bit enables the relative frame counter 2 reload interrupt generation.  
0: Relative frame counter 2 reload interrupt disabled  
1: Relative frame counter 2 reload interrupt enabled  

Bit 12  **RFC1RIE**: Relative frame counter 1 reload interrupt enable  
This bit enables the relative frame counter 1 reload interrupt generation.  
0: Relative frame counter 1 reload interrupt disabled  
1: Relative frame counter 1 reload interrupt enabled  

Bits 11:10 Reserved, must be kept at reset value.  

Bit 9  **ALCC2IE**: Absolute line counter compare 2 interrupt enable  
This bit enables the absolute line counter compare 2 interrupt generation.  
0: Absolute line counter compare 2 interrupt disabled  
1: Absolute line counter compare 2 interrupt enabled  

Bit 8  **ALCC1IE**: Absolute line counter compare 1 interrupt enable  
This bit enables the absolute line counter compare 1 interrupt generation.  
0: Absolute line counter compare 1 interrupt disabled  
1: Absolute line counter compare 1 interrupt enabled  

Bits 7:5 Reserved, must be kept at reset value.  

Bit 4  **AFCC1IE**: Absolute frame counter compare 1 interrupt enable  
This bit enables the absolute frame counter compare 1 interrupt generation.  
0: Absolute frame counter compare 1 interrupt disabled  
1: Absolute frame counter compare 1 interrupt enabled  

Bit 3 Reserved, must be kept at reset value.  

Bit 2  **TEIE**: Tearing-effect interrupt enable  
This bit enables the Tearing Effect interrupt generation.  
0: Tearing-effect interrupt disabled  
1: Tearing-effect interrupt enabled  

Bit 1  **ALCOIE**: Absolute line counter overflow interrupt enable  
This bit enables the absolute line counter overflow interrupt generation.  
0: Absolute line counter overflow interrupt disabled  
1: Absolute line counter overflow interrupt enabled
Bit 0  **AFCOIE**: Absolute frame counter overflow interrupt enable
This bit enables the absolute frame counter overflow interrupt generation.
0: Absolute frame counter overflow interrupt disabled
1: Absolute frame counter overflow interrupt enabled

18.5.11  **GFXTIM timers status register (GFXTIM_TSR)**

Address offset: 0x03C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

Bits 31:21  Reserved, must be kept at reset value.

Bit 20  **RFC2S**: Relative frame counter 2 status
This bit returns the status of the relative frame counter 2.
0: Relative frame counter 2 disabled
1: Relative frame counter 2 enabled

Bits 19:17  Reserved, must be kept at reset value.

Bit 16  **RFC1S**: Relative frame counter 1 status
This bit returns the status of the relative frame counter 1.
0: Relative frame counter 1 disabled
1: Relative frame counter 1 enabled

Bits 15:5  Reserved, must be kept at reset value.

Bit 4  **ALCS**: Absolute line counter status
This bit returns the status of the absolute line counter.
0: Absolute line counter disabled
1: Absolute line counter enabled

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **AFCS**: Absolute frame counter status
This bit returns the status of the absolute frame counter.
0: Absolute frame counter disabled
1: Absolute frame counter enabled
### 18.5.12 GFXTIM line clock counter reload register (GFXTIM_LCCRR)

Address offset: 0x040  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:22 Reserved, must be kept at reset value.  
Bits 21:0 **RELOAD[21:0]**: Reload value  
Reload value of the line clock counter.

### 18.5.13 GFXTIM frame clock counter reload register (GFXTIM_FCCRR)

Address offset: 0x044  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.  
Bits 11:0 **RELOAD[11:0]**: Reload value  
Reload value of the frame clock counter.

### 18.5.14 GFXTIM absolute time register (GFXTIM_ATR)

Address offset: 0x050  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12 **FRAME[19:0]**: Frame number  
Current value of the absolute frame counter.
18.5.15  **GFXTIM absolute frame counter register (GFXTIM_AFCR)**

Address offset: 0x054

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FRAME[19:16]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 11:0**  **LINE[11:0]: Line number**

Current value of the absolute line counter.

**Note:** This bitfield can only be written when the absolute line counter is disabled.

18.5.16  **GFXTIM absolute line counter register (GFXTIM_ALCR)**

Address offset: 0x058

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:12**  **Reserved, must be kept at reset value.**

**Bits 11:0**  **LINE[11:0]: Line number**

Current value of the absolute line counter.

**Note:** This bitfield can only be written when the absolute line counter is disabled.
### 18.5.17 GFXTIM absolute frame counter compare 1 register (GFXTIM_AFCC1R)

- **Address offset:** 0x060
- **Reset value:** 0x0000 0000

#### Bits 31:20
Reserved, must be kept at reset value.

#### Bits 19:0
- **FRAME[19:0]:** Frame number
  - Compare 1 value for the absolute frame counter.

### 18.5.18 GFXTIM absolute line counter compare 1 register (GFXTIM_ALCC1R)

- **Address offset:** 0x070
- **Reset value:** 0x0000 0000

#### Bits 31:12
Reserved, must be kept at reset value.

#### Bits 11:0
- **LINE[11:0]:** Line number
  - Compare value 1 for the absolute line counter.

### 18.5.19 GFXTIM absolute line counter compare 2 register (GFXTIM_ALCC2R)

- **Address offset:** 0x074
- **Reset value:** 0x0000 0000

#### Bits 31:12
Reserved, must be kept at reset value.

#### Bits 11:0
- **LINE[11:0]:** Line number
  - Compare value 1 for the absolute line counter.
18.5.20  GFXTIM relative frame counter 1 register (GFXTIM_RFC1R)

Address offset: 0x080
Reset value: 0x0000 0000

Bits 31:12  Reserved, must be kept at reset value.
Bits 11:0  **LINE[11:0]:** Line number
Compare value 2 for the absolute line counter.

18.5.21  GFXTIM relative frame counter 1 reload register (GFXTIM_RFC1RR)

Address offset: 0x084
Reset value: 0x0000 0000

Bits 31:12  Reserved, must be kept at reset value.
Bits 11:0  **FRAME[11:0]:** Frame number
Current value of the relative frame counter 1.
### 18.5.22 GFXTIM relative frame counter 2 register (GFXTIM_RFC2R)

Address offset: 0x088  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:12 Reserved, must be kept at reset value.  
Bits 11:0 **FRAME[11:0]**: Frame number  
Current value of the relative frame counter 2.

### 18.5.23 GFXTIM relative frame counter 2 reload register (GFXTIM_RFC2RR)

Address offset: 0x08C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:12 Reserved, must be kept at reset value.  
Bits 11:0 **FRAME[11:0]**: Frame reload value  
Reload value for the relative frame counter 2.

### 18.5.24 GFXTIM watchdog counter register (GFXTIM_WDGCR)

Address offset: 0x0A0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>VALUE[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.
Bits 15:0 \textbf{VALUE}[15:0]: Value
Current value of the watchdog counter.

18.5.25 **GFXTIM watchdog reload register (GFXTIM\_WDGRR)**
Address offset: 0x0A4
Reset value: 0x0000 0000

```
| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | \textbf{RELOAD}[15:0]: Reload value |
|           | Reload value of the watchdog counter. |
```

18.5.26 **GFXTIM watchdog pre-alarm register (GFXTIM\_WDGPAR)**
Address offset: 0x0A8
Reset value: 0x0000 0000

```
| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | \textbf{PREALARM}[15:0]: Pre-alarm value |
|           | Pre-alarm value of the watchdog counter. |
```

18.5.27 **GFXTIM register map**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>GFXTIM_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>GFXTIM_CGCR</td>
<td>FCHR[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 141. GFXTIM register map and reset values
### Table 141. GFXTIM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Name 31-29</th>
<th>Name 28-27</th>
<th>26-24</th>
<th>23-22</th>
<th>21-20</th>
<th>19-18</th>
<th>17-16</th>
<th>15-14</th>
<th>13-12</th>
<th>11-10</th>
<th>9-8</th>
<th>7-6</th>
<th>5-4</th>
<th>3-2</th>
<th>1-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x008</td>
<td>GFXTIM_TCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>GFXTIM_TDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>GFXTIM_EVCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018-</td>
<td>GFXTIM_WDGTCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>�</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>GFXTIM_ISR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024-</td>
<td>GFXTIM_ICR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>GFXTIM_IER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>GFXTIM_TSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>GFXTIM_FCCRR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x054</td>
<td>GFXTIM_AFCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x058</td>
<td>GFXTIM_ALCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x05C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Table 141. GFXTIM register map and reset values (continued)

| Offset | Register name  | Name | Offset | Name | Offset | Name | Offset | Name | Offset | Name | Offset | Name | Offset | Name |
|--------|----------------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|--------|------|
| 0x060  | GFXTIM_AFC1R   |      | 31     |      | 30     |      | 29     |      | 28     |      | 27     |      | 26     |      | 25     |      |
|        |                |      | 24     |      | 23     |      | 22     |      | 21     |      | 20     |      | 19     |      | 18     |      |
|        |                |      | 17     |      | 16     |      | 15     |      | 14     |      | 13     |      | 12     |      | 11     |      |
|        |                |      | 10     |      | 9      |      | 8      |      | 7      |      | 6      |      | 5      |      | 4      |      |
|        |                |      | 3       |      | 2      |      | 1      |      | 0      |      |
| 0x064-0x06C | Reserved |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x070  | GFXTIM_ALCC1R  |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x074  | GFXTIM_ALCC2R  |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x078-0x07C | Reserved |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x080  | GFXTIM_RFC1R   |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x084  | GFXTIM_RFC1RR  |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x088  | GFXTIM_RFC2R   |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x08C  | GFXTIM_RFC2RR  |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x090-0x09C | Reserved |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x0A0  | GFXTIM_WDGCCR  |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x0A4  | GFXTIM_WDGCR   |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
| 0x0A8  | GFXTIM_WDPAR   |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |
|        |                |      |        |      |        |      |        |      |        |      |        |      |        |      |        |      |

Refer to Section 2.3 for the register boundary addresses.
19 Nested vectored interrupt controller (NVIC)

19.1 NVIC features

The NVIC includes the following features:

- up to 140 maskable interrupt channels for STM32H7xxx (not including the 10 interrupt lines of Cortex®-M7 with FPU)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- low-latency exception and interrupt handling
- power management control
- implementation of system control registers

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

All interrupts, including the core exceptions, are managed by the NVIC.

For more information on exceptions and NVIC programming, refer to PM0253 programming manual for Cortex®-M7.

19.1.1 SysTick calibration value register

The SysTick calibration value (SYST_CALIB) is fixed to 0x3E8. It provides a reference timebase of 1 ms based when the SysTick clock frequency is 1 MHz. To match the 1 ms timebase whatever the application frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- If the SysTick clock source is the 100 MHz CPU clock (HCLK):
  \[
  \text{reload value} = (F_{\text{HCLK}} \times \text{SYST_CALIB}) - 1
  \]

- If the SysTick clock source is an external clock:
  \[
  \text{reload value} = \left(\frac{F_{\text{HCLK}}}{8}\right) \times \text{SYST_CALIB} - 1
  \]

where \( F_{\text{HCLK}} \) refers to the AHB frequency expressed in MHz.

For example, to achieve a timebase of 1 ms when the SysTick clock source is the 100 MHz HCLK:

\[
\text{reload value} = (100 \times \text{SYST_CALIB}) - 1 = 0x1869F
\]
### 19.1.2 Interrupt and exception vectors

The exception vectors connected to the NVIC are the following: reset, NMI, HardFault, MemManage, Bus Fault, UsageFault, SVCall, DebugMonitor, PendSV, SysTick.

<table>
<thead>
<tr>
<th>NVIC position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td></td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>-</td>
<td>-3</td>
<td>fixed</td>
<td>Reset</td>
<td>Non maskable interrupt. The RCC Clock Security System (HSE CSS) is linked to the NMI vector.</td>
<td>0x0000 0004</td>
</tr>
<tr>
<td>-</td>
<td>-2</td>
<td>fixed</td>
<td>NMI</td>
<td></td>
<td>0x0000 0008</td>
</tr>
<tr>
<td>-</td>
<td>-1</td>
<td>fixed</td>
<td>HardFault</td>
<td>All class of fault</td>
<td>0x0000 000C</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>fixed</td>
<td>MemManage</td>
<td>Memory management</td>
<td>0x0000 0010</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>settable</td>
<td>BusFault</td>
<td>Pre-fetch fault, memory access fault</td>
<td>0x0000 0014</td>
</tr>
<tr>
<td>-</td>
<td>2</td>
<td>settable</td>
<td>UsageFault</td>
<td>Undefined instruction or illegal state</td>
<td>0x0000 0018</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td></td>
<td>0x0000 001C</td>
</tr>
<tr>
<td>-</td>
<td>3</td>
<td>settable</td>
<td>SVCall</td>
<td>System service call via SWI instruction</td>
<td>0x0000 002C</td>
</tr>
<tr>
<td>-</td>
<td>4</td>
<td>settable</td>
<td>Debug Monitor</td>
<td>Debug Monitor</td>
<td>0x0000 0030</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x0000 0034</td>
</tr>
<tr>
<td>-</td>
<td>5</td>
<td>settable</td>
<td>PendSV</td>
<td>Pendable request for system service</td>
<td>0x0000 0038</td>
</tr>
<tr>
<td>-</td>
<td>6</td>
<td>settable</td>
<td>Systick</td>
<td>System tick timer</td>
<td>0x0000 003C</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>settable</td>
<td>PVD_AVD</td>
<td>PVD and AVD through the EXTI line</td>
<td>0x0000 0040</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>0x0000 0044</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>settable</td>
<td>DTS</td>
<td>DTS global interrupt</td>
<td>0x0000 0048</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>settable</td>
<td>IWDG</td>
<td>Independent watchdog interrupt</td>
<td>0x0000 004C</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>settable</td>
<td>WWDG</td>
<td>Window watchdog interrupt</td>
<td>0x0000 0050</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>settable</td>
<td>RCC</td>
<td>RCC global interrupt</td>
<td>0x0000 0054</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>settable</td>
<td>LOOKUP</td>
<td>LOOKUP/Overstack, …</td>
<td>0x0000 0058</td>
</tr>
<tr>
<td>7</td>
<td>14</td>
<td>settable</td>
<td>CACHE_ECC</td>
<td>Error ECC cache</td>
<td>0x0000 005C</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>settable</td>
<td>FLASH</td>
<td>Flash memory interface</td>
<td>0x0000 0060</td>
</tr>
</tbody>
</table>
### Table 142. NVIC\(^{(1)}\)

<table>
<thead>
<tr>
<th>NVIC position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>16</td>
<td>settable</td>
<td>ECC_FPU</td>
<td>ECC/FPU/All flag from exec</td>
<td>0x0000 0064</td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>settable</td>
<td>FPU</td>
<td>FPU safety Flag</td>
<td>0x0000 0068</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 006C</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 0070</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>settable</td>
<td>TAMP</td>
<td>RTC tamper and timestamp interrupts through the EXTI line</td>
<td>0x0000 0074</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved (product secure)</td>
<td>0x0000 0078</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved (product secure)</td>
<td>0x0000 007C</td>
</tr>
<tr>
<td>16</td>
<td>23</td>
<td>settable</td>
<td>EXTI0</td>
<td>EXTI Line 0 interrupt through the EXTI line</td>
<td>0x0000 0080</td>
</tr>
<tr>
<td>17</td>
<td>24</td>
<td>settable</td>
<td>EXTI1</td>
<td>EXTI Line 1 interrupt through the EXTI line</td>
<td>0x0000 0084</td>
</tr>
<tr>
<td>18</td>
<td>25</td>
<td>settable</td>
<td>EXTI2</td>
<td>EXTI Line 2 interrupt through the EXTI line</td>
<td>0x0000 0088</td>
</tr>
<tr>
<td>19</td>
<td>26</td>
<td>settable</td>
<td>EXTI3</td>
<td>EXTI Line 3 interrupt through the EXTI line</td>
<td>0x0000 008C</td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>settable</td>
<td>EXTI4</td>
<td>EXTI Line 4 interrupt through the EXTI line</td>
<td>0x0000 0090</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>settable</td>
<td>EXTI5</td>
<td>EXTI Line 5 interrupt through the EXTI line</td>
<td>0x0000 0094</td>
</tr>
<tr>
<td>22</td>
<td>29</td>
<td>settable</td>
<td>EXTI6</td>
<td>EXTI Line 6 interrupt through the EXTI line</td>
<td>0x0000 0098</td>
</tr>
<tr>
<td>23</td>
<td>30</td>
<td>settable</td>
<td>EXTI7</td>
<td>EXTI Line 7 interrupt through the EXTI line</td>
<td>0x0000 009C</td>
</tr>
<tr>
<td>24</td>
<td>31</td>
<td>settable</td>
<td>EXTI8</td>
<td>EXTI Line 8 interrupt through the EXTI line</td>
<td>0x0000 00A0</td>
</tr>
<tr>
<td>25</td>
<td>32</td>
<td>settable</td>
<td>EXTI9</td>
<td>EXTI Line 9 interrupt through the EXTI line</td>
<td>0x0000 00A4</td>
</tr>
<tr>
<td>26</td>
<td>33</td>
<td>settable</td>
<td>EXTI10</td>
<td>EXTI Line 10 interrupt through the EXTI line</td>
<td>0x0000 00A8</td>
</tr>
<tr>
<td>27</td>
<td>34</td>
<td>settable</td>
<td>EXTI11</td>
<td>EXTI Line 11 interrupt through the EXTI line</td>
<td>0x0000 00AC</td>
</tr>
<tr>
<td>28</td>
<td>35</td>
<td>settable</td>
<td>EXTI12</td>
<td>EXTI Line 12 interrupt through the EXTI line</td>
<td>0x0000 00B0</td>
</tr>
<tr>
<td>29</td>
<td>36</td>
<td>settable</td>
<td>EXTI13</td>
<td>EXTI Line 13 interrupt through the EXTI line</td>
<td>0x0000 00B4</td>
</tr>
<tr>
<td>30</td>
<td>37</td>
<td>settable</td>
<td>EXTI14</td>
<td>EXTI Line 14 interrupt through the EXTI line</td>
<td>0x0000 00B8</td>
</tr>
<tr>
<td>31</td>
<td>38</td>
<td>settable</td>
<td>EXTI15</td>
<td>EXTI Line 15 interrupt through the EXTI line</td>
<td>0x0000 00BC</td>
</tr>
<tr>
<td>32</td>
<td>39</td>
<td>settable</td>
<td>RTC</td>
<td>RTC Wakeup and Alarm interrupt through the EXTI line</td>
<td>0x0000 00C0</td>
</tr>
<tr>
<td>33</td>
<td>40</td>
<td>settable</td>
<td>SAES</td>
<td>SAES global interrupt</td>
<td>0x0000 00C4</td>
</tr>
<tr>
<td>34</td>
<td>41</td>
<td>settable</td>
<td>AES</td>
<td>AES global interrupt</td>
<td>0x0000 00C8</td>
</tr>
<tr>
<td>35</td>
<td>42</td>
<td>settable</td>
<td>PKA</td>
<td>PKA global interrupt</td>
<td>0x0000 00CC</td>
</tr>
<tr>
<td>NVIC position</td>
<td>Priority</td>
<td>Type of priority</td>
<td>Acronym</td>
<td>Description</td>
<td>Address offset</td>
</tr>
<tr>
<td>---------------</td>
<td>----------</td>
<td>------------------</td>
<td>---------------</td>
<td>------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>36</td>
<td>43</td>
<td>settable</td>
<td>HASH</td>
<td>HASH global interrupt</td>
<td>0x0000 00D0</td>
</tr>
<tr>
<td>37</td>
<td>44</td>
<td>settable</td>
<td>RNG</td>
<td>RNG global interrupt</td>
<td>0x0000 00D4</td>
</tr>
<tr>
<td>38</td>
<td>45</td>
<td>settable</td>
<td>ADC1_2</td>
<td>ADC1/2 global interrupt</td>
<td>0x0000 00D8</td>
</tr>
<tr>
<td>39</td>
<td>46</td>
<td>settable</td>
<td>GPDMA1_CH0</td>
<td>GPDMA1 channel 0 interrupt</td>
<td>0x0000 00DC</td>
</tr>
<tr>
<td>40</td>
<td>47</td>
<td>settable</td>
<td>GPDMA1_CH1</td>
<td>GPDMA1 channel 1 interrupt</td>
<td>0x0000 00E0</td>
</tr>
<tr>
<td>41</td>
<td>48</td>
<td>settable</td>
<td>GPDMA1_CH2</td>
<td>GPDMA1 channel 2 interrupt</td>
<td>0x0000 00E4</td>
</tr>
<tr>
<td>42</td>
<td>49</td>
<td>settable</td>
<td>GPDMA1_CH3</td>
<td>GPDMA1 channel 3 interrupt</td>
<td>0x0000 00E8</td>
</tr>
<tr>
<td>43</td>
<td>50</td>
<td>settable</td>
<td>GPDMA1_CH4</td>
<td>GPDMA1 channel 4 interrupt</td>
<td>0x0000 00EC</td>
</tr>
<tr>
<td>44</td>
<td>51</td>
<td>settable</td>
<td>GPDMA1_CH5</td>
<td>GPDMA1 channel 5 interrupt</td>
<td>0x0000 00F0</td>
</tr>
<tr>
<td>45</td>
<td>52</td>
<td>settable</td>
<td>GPDMA1_CH6</td>
<td>GPDMA1 channel 6 interrupt</td>
<td>0x0000 00F4</td>
</tr>
<tr>
<td>46</td>
<td>53</td>
<td>settable</td>
<td>GPDMA1_CH7</td>
<td>GPDMA1 channel 7 interrupt</td>
<td>0x0000 00F8</td>
</tr>
<tr>
<td>47</td>
<td>54</td>
<td>settable</td>
<td>TIM1_BRK</td>
<td>TIM1 Break interrupt (tim_brk_terr_ierr_it)</td>
<td>0x0000 00FC</td>
</tr>
<tr>
<td>48</td>
<td>55</td>
<td>settable</td>
<td>TIM1_UP</td>
<td>TIM1 Update interrupt (tim_upd_it)</td>
<td>0x0000 0100</td>
</tr>
<tr>
<td>49</td>
<td>56</td>
<td>settable</td>
<td>TIM1_TRG_COM</td>
<td>TIM1 Trigger and Commutation interrupts (tim_trg_com_dir_idx_it)</td>
<td>0x0000 0104</td>
</tr>
<tr>
<td>50</td>
<td>57</td>
<td>settable</td>
<td>TIM1_CC</td>
<td>TIM1 Capture Compare interrupt (tim_cc_it)</td>
<td>0x0000 0108</td>
</tr>
<tr>
<td>51</td>
<td>58</td>
<td>settable</td>
<td>TIM2</td>
<td>TIM2 global interrupt</td>
<td>0x0000 010C</td>
</tr>
<tr>
<td>52</td>
<td>59</td>
<td>settable</td>
<td>TIM3</td>
<td>TIM3 global interrupt</td>
<td>0x0000 0110</td>
</tr>
<tr>
<td>53</td>
<td>60</td>
<td>settable</td>
<td>TIM4</td>
<td>TIM4 global interrupt</td>
<td>0x0000 0114</td>
</tr>
<tr>
<td>54</td>
<td>61</td>
<td>settable</td>
<td>TIM5</td>
<td>TIM5 global interrupt</td>
<td>0x0000 0118</td>
</tr>
<tr>
<td>55</td>
<td>62</td>
<td>settable</td>
<td>TIM6</td>
<td>TIM6 global interrupt</td>
<td>0x0000 011C</td>
</tr>
<tr>
<td>56</td>
<td>63</td>
<td>settable</td>
<td>TIM7</td>
<td>TIM7 global interrupt</td>
<td>0x0000 0120</td>
</tr>
<tr>
<td>57</td>
<td>64</td>
<td>settable</td>
<td>TIM9</td>
<td>TIM9 global interrupt</td>
<td>0x0000 0124</td>
</tr>
<tr>
<td>58</td>
<td>65</td>
<td>settable</td>
<td>SPI1</td>
<td>SPI1 global interrupt</td>
<td>0x0000 0128</td>
</tr>
<tr>
<td>59</td>
<td>66</td>
<td>settable</td>
<td>SPI2</td>
<td>SPI2 global interrupt</td>
<td>0x0000 012C</td>
</tr>
<tr>
<td>60</td>
<td>67</td>
<td>settable</td>
<td>SPI3</td>
<td>SPI3 global interrupt</td>
<td>0x0000 0130</td>
</tr>
<tr>
<td>61</td>
<td>68</td>
<td>settable</td>
<td>SPI4</td>
<td>SPI4 global interrupt</td>
<td>0x0000 0134</td>
</tr>
<tr>
<td>62</td>
<td>69</td>
<td>settable</td>
<td>SPI5</td>
<td>SPI5 global interrupt</td>
<td>0x0000 0138</td>
</tr>
<tr>
<td>63</td>
<td>70</td>
<td>settable</td>
<td>SPI6</td>
<td>SPI6 global interrupt</td>
<td>0x0000 013C</td>
</tr>
</tbody>
</table>
## Table 142. NVIC(1)

<table>
<thead>
<tr>
<th>NVIC position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>71</td>
<td>settable</td>
<td>HPDMA1_CH0</td>
<td>HPDMA1 channel 0 interrupt</td>
<td>0x0000 0140</td>
</tr>
<tr>
<td>65</td>
<td>72</td>
<td>settable</td>
<td>HPDMA1_CH1</td>
<td>HPDMA1 channel 1 interrupt</td>
<td>0x0000 0144</td>
</tr>
<tr>
<td>66</td>
<td>73</td>
<td>settable</td>
<td>HPDMA1_CH2</td>
<td>HPDMA1 channel 2 interrupt</td>
<td>0x0000 0148</td>
</tr>
<tr>
<td>67</td>
<td>74</td>
<td>settable</td>
<td>HPDMA1_CH3</td>
<td>HPDMA1 channel 3 interrupt</td>
<td>0x0000 014C</td>
</tr>
<tr>
<td>68</td>
<td>75</td>
<td>settable</td>
<td>HPDMA1_CH4</td>
<td>HPDMA1 channel 4 interrupt</td>
<td>0x0000 0150</td>
</tr>
<tr>
<td>69</td>
<td>76</td>
<td>settable</td>
<td>HPDMA1_CH5</td>
<td>HPDMA1 channel 5 interrupt</td>
<td>0x0000 0154</td>
</tr>
<tr>
<td>70</td>
<td>77</td>
<td>settable</td>
<td>HPDMA1_CH6</td>
<td>HPDMA1 channel 6 interrupt</td>
<td>0x0000 0158</td>
</tr>
<tr>
<td>71</td>
<td>78</td>
<td>settable</td>
<td>HPDMA1_CH7</td>
<td>HPDMA1 channel 7 interrupt</td>
<td>0x0000 015C</td>
</tr>
<tr>
<td>72</td>
<td>80</td>
<td>settable</td>
<td>SAI1_A</td>
<td>SAI1 global interrupt A</td>
<td>0x0000 0160</td>
</tr>
<tr>
<td>73</td>
<td>81</td>
<td>settable</td>
<td>SAI1_B</td>
<td>SAI1 global interrupt B</td>
<td>0x0000 0164</td>
</tr>
<tr>
<td>74</td>
<td>82</td>
<td>settable</td>
<td>SAI2_A</td>
<td>SAI2 global interrupt A</td>
<td>0x0000 0168</td>
</tr>
<tr>
<td>75</td>
<td>83</td>
<td>settable</td>
<td>SAI2_B</td>
<td>SAI2 global interrupt B</td>
<td>0x0000 016C</td>
</tr>
<tr>
<td>76</td>
<td>84</td>
<td>settable</td>
<td>I2C1_EV</td>
<td>I2C1 event interrupt</td>
<td>0x0000 0170</td>
</tr>
<tr>
<td>77</td>
<td>85</td>
<td>settable</td>
<td>I2C1_ER</td>
<td>I2C1 error interrupt</td>
<td>0x0000 0174</td>
</tr>
<tr>
<td>78</td>
<td>86</td>
<td>settable</td>
<td>I2C2_EV</td>
<td>I2C2 event interrupt</td>
<td>0x0000 0178</td>
</tr>
<tr>
<td>79</td>
<td>87</td>
<td>settable</td>
<td>I2C2_ER</td>
<td>I2C2 error interrupt</td>
<td>0x0000 017C</td>
</tr>
<tr>
<td>80</td>
<td>88</td>
<td>settable</td>
<td>I2C3_EV</td>
<td>I2C3 event interrupt</td>
<td>0x0000 0180</td>
</tr>
<tr>
<td>81</td>
<td>89</td>
<td>settable</td>
<td>I2C3_ER</td>
<td>I2C3 error interrupt</td>
<td>0x0000 0184</td>
</tr>
<tr>
<td>82</td>
<td>90</td>
<td>settable</td>
<td>USART1</td>
<td>USART1 global interrupt</td>
<td>0x0000 0188</td>
</tr>
<tr>
<td>83</td>
<td>91</td>
<td>settable</td>
<td>USART2</td>
<td>USART2 global interrupt</td>
<td>0x0000 018C</td>
</tr>
<tr>
<td>84</td>
<td>92</td>
<td>settable</td>
<td>USART3</td>
<td>USART3 global interrupt</td>
<td>0x0000 0190</td>
</tr>
<tr>
<td>85</td>
<td>93</td>
<td>settable</td>
<td>UART4</td>
<td>UART4 global interrupt</td>
<td>0x0000 0194</td>
</tr>
<tr>
<td>86</td>
<td>94</td>
<td>settable</td>
<td>UART5</td>
<td>UART5 global interrupt</td>
<td>0x0000 0198</td>
</tr>
<tr>
<td>87</td>
<td>95</td>
<td>settable</td>
<td>UART7</td>
<td>UART7 global interrupt</td>
<td>0x0000 019C</td>
</tr>
<tr>
<td>88</td>
<td>96</td>
<td>settable</td>
<td>UART8</td>
<td>UART8 global interrupt</td>
<td>0x0000 01A0</td>
</tr>
<tr>
<td>89</td>
<td>97</td>
<td>settable</td>
<td>I3C1_EV</td>
<td>I3C1 event interrupt</td>
<td>0x0000 01A4</td>
</tr>
<tr>
<td>90</td>
<td>98</td>
<td>settable</td>
<td>I3C1_ER</td>
<td>I3C1 error interrupt</td>
<td>0x0000 01A8</td>
</tr>
<tr>
<td>91</td>
<td>99</td>
<td>settable</td>
<td>OTG_HS</td>
<td>USB OTG HS global interrupt</td>
<td>0x0000 01AC</td>
</tr>
<tr>
<td>92</td>
<td></td>
<td>settable</td>
<td>ETH</td>
<td>Ethernet global interrupt</td>
<td>0x0000 01B0</td>
</tr>
</tbody>
</table>
### Table 142. NVIC(1)

<table>
<thead>
<tr>
<th>NVIC position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>93</td>
<td>100</td>
<td>settable</td>
<td>CORDIC</td>
<td>CORDIC interrupt</td>
<td>0x0000 01B4</td>
</tr>
<tr>
<td>94</td>
<td>101</td>
<td>settable</td>
<td>GFXTIM</td>
<td>GFXTIM global interrupt</td>
<td>0x0000 01B8</td>
</tr>
<tr>
<td>95</td>
<td>102</td>
<td>settable</td>
<td>DCMIPP</td>
<td>DCMI global interrupt</td>
<td>0x0000 01BC</td>
</tr>
<tr>
<td>96</td>
<td>103</td>
<td>settable</td>
<td>LTDC</td>
<td>LCD global interrupt</td>
<td>0x0000 01C0</td>
</tr>
<tr>
<td>97</td>
<td>104</td>
<td>settable</td>
<td>LTDC_ER</td>
<td>LCD error interrupt</td>
<td>0x0000 01C4</td>
</tr>
<tr>
<td>98</td>
<td>105</td>
<td>settable</td>
<td>DMA2D</td>
<td>DMA2D global interrupt</td>
<td>0x0000 01C8</td>
</tr>
<tr>
<td>99</td>
<td>106</td>
<td>settable</td>
<td>JPEG</td>
<td>JPEG global interrupt</td>
<td>0x0000 01CC</td>
</tr>
<tr>
<td>100</td>
<td>107</td>
<td>settable</td>
<td>GFXMMU</td>
<td>GFXMMU global interrupt</td>
<td>0x0000 01D0</td>
</tr>
<tr>
<td>101</td>
<td>108</td>
<td>settable</td>
<td>I3C1_WKUP</td>
<td>I3C wakeup Interrupt through EXTI line</td>
<td>0x0000 01D4</td>
</tr>
<tr>
<td>102</td>
<td>109</td>
<td>settable</td>
<td>MCE1</td>
<td>MCE1 global interrupt</td>
<td>0x0000 01D8</td>
</tr>
<tr>
<td>103</td>
<td>110</td>
<td>settable</td>
<td>MCE2</td>
<td>MCE2 global interrupt</td>
<td>0x0000 01DC</td>
</tr>
<tr>
<td>104</td>
<td>111</td>
<td>settable</td>
<td>MCE3</td>
<td>MCE3 global interrupt</td>
<td>0x0000 01E0</td>
</tr>
<tr>
<td>105</td>
<td>112</td>
<td>settable</td>
<td>OSPI1</td>
<td>OSPI1 / HSPI1 global interrupt</td>
<td>0x0000 01E4</td>
</tr>
<tr>
<td>106</td>
<td>113</td>
<td>settable</td>
<td>OSPI2</td>
<td>OSPI2 global interrupt</td>
<td>0x0000 01E8</td>
</tr>
<tr>
<td>107</td>
<td>114</td>
<td>settable</td>
<td>FMC</td>
<td>FMC global interrupt</td>
<td>0x0000 01EC</td>
</tr>
<tr>
<td>108</td>
<td>115</td>
<td>settable</td>
<td>SDMMC1</td>
<td>SDMMC1 global interrupt</td>
<td>0x0000 01F0</td>
</tr>
<tr>
<td>109</td>
<td>116</td>
<td>settable</td>
<td>SDMMC2</td>
<td>SDMMC2 global interrupt</td>
<td>0x0000 01F4</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01F8</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
<td>0x0000 01FC</td>
</tr>
<tr>
<td>112</td>
<td>119</td>
<td>settable</td>
<td>OTG_FS</td>
<td>USB OTG FS global interrupt</td>
<td>0x0000 0200</td>
</tr>
<tr>
<td>113</td>
<td>120</td>
<td>settable</td>
<td>TIM12</td>
<td>TIM12 global interrupt</td>
<td>0x0000 0204</td>
</tr>
<tr>
<td>114</td>
<td>121</td>
<td>settable</td>
<td>TIM13</td>
<td>TIM13 global interrupt</td>
<td>0x0000 0208</td>
</tr>
<tr>
<td>115</td>
<td>122</td>
<td>settable</td>
<td>TIM14</td>
<td>TIM14 global interrupt</td>
<td>0x0000 020C</td>
</tr>
<tr>
<td>116</td>
<td>123</td>
<td>settable</td>
<td>TIM15</td>
<td>TIM15 global interrupt</td>
<td>0x0000 0210</td>
</tr>
<tr>
<td>117</td>
<td>124</td>
<td>settable</td>
<td>TIM16</td>
<td>TIM16 global interrupt</td>
<td>0x0000 0214</td>
</tr>
<tr>
<td>118</td>
<td>125</td>
<td>settable</td>
<td>TIM17</td>
<td>TIM17 global interrupt</td>
<td>0x0000 0218</td>
</tr>
<tr>
<td>119</td>
<td>126</td>
<td>settable</td>
<td>LPTIM1</td>
<td>LPTIM1 global interrupt</td>
<td>0x0000 021C</td>
</tr>
<tr>
<td>120</td>
<td>127</td>
<td>settable</td>
<td>LPTIM2</td>
<td>LPTIM2 global interrupt</td>
<td>0x0000 0220</td>
</tr>
<tr>
<td>121</td>
<td>128</td>
<td>settable</td>
<td>LPTIM3</td>
<td>LPTIM3 global interrupt</td>
<td>0x0000 0224</td>
</tr>
</tbody>
</table>
Table 142. NVIC\(^{1}\)

<table>
<thead>
<tr>
<th>NVIC position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>122</td>
<td>129</td>
<td>settable</td>
<td>LPTIM4</td>
<td>LPTIM4 global interrupt</td>
<td>0x0000 0228</td>
</tr>
<tr>
<td>123</td>
<td>130</td>
<td>settable</td>
<td>LPTIM5</td>
<td>LPTIM5 global interrupt</td>
<td>0x0000 022C</td>
</tr>
<tr>
<td>124</td>
<td>131</td>
<td>settable</td>
<td>SPDIF_RX</td>
<td>SPDIFRX global interrupt</td>
<td>0x0000 0230</td>
</tr>
<tr>
<td>125</td>
<td>132</td>
<td>settable</td>
<td>MDIOS</td>
<td>MDIOS global interrupt</td>
<td>0x0000 0234</td>
</tr>
<tr>
<td>126</td>
<td>133</td>
<td>settable</td>
<td>ADF1FLT0</td>
<td>ADF1 filter 0 global interrupt</td>
<td>0x0000 0238</td>
</tr>
<tr>
<td>127</td>
<td>134</td>
<td>settable</td>
<td>CRS</td>
<td>CRS global interrupt</td>
<td>0x0000 023C</td>
</tr>
<tr>
<td>128</td>
<td>135</td>
<td>settable</td>
<td>UCPD1</td>
<td>UCPD global interrupt</td>
<td>0x0000 0240</td>
</tr>
<tr>
<td>129</td>
<td>136</td>
<td>settable</td>
<td>CEC</td>
<td>CEC global interrupt</td>
<td>0x0000 0244</td>
</tr>
<tr>
<td>130</td>
<td>137</td>
<td>settable</td>
<td>UCPD1</td>
<td>UCPD global interrupt</td>
<td>0x0000 0248</td>
</tr>
<tr>
<td>131</td>
<td>138</td>
<td>settable</td>
<td>LPUART1</td>
<td>LPUART1 global interrupt</td>
<td>0x0000 024C</td>
</tr>
<tr>
<td>132</td>
<td>139</td>
<td>settable</td>
<td>WAKEUP_PIN</td>
<td>Interrupt for 4 wakeup pins (1, 2, 3, 4) through EXTI line</td>
<td>0x0000 0250</td>
</tr>
<tr>
<td>133</td>
<td>140</td>
<td>settable</td>
<td>GPDMA1_CH8</td>
<td>GPDMA1 channel 8 interrupt</td>
<td>0x0000 0254</td>
</tr>
<tr>
<td>134</td>
<td>141</td>
<td>settable</td>
<td>GPDMA1_CH9</td>
<td>GPDMA1 channel 9 interrupt</td>
<td>0x0000 0258</td>
</tr>
<tr>
<td>135</td>
<td>142</td>
<td>settable</td>
<td>GPDMA1_CH10</td>
<td>GPDMA1 channel 10 interrupt</td>
<td>0x0000 025C</td>
</tr>
<tr>
<td>136</td>
<td>143</td>
<td>settable</td>
<td>GPDMA1_CH11</td>
<td>GPDMA1 channel 11 interrupt</td>
<td>0x0000 0260</td>
</tr>
<tr>
<td>137</td>
<td>144</td>
<td>settable</td>
<td>GPDMA1_CH12</td>
<td>GPDMA1 channel 12 interrupt</td>
<td>0x0000 0264</td>
</tr>
<tr>
<td>138</td>
<td>145</td>
<td>settable</td>
<td>GPDMA1_CH13</td>
<td>GPDMA1 channel 13 interrupt</td>
<td>0x0000 0268</td>
</tr>
<tr>
<td>139</td>
<td>146</td>
<td>settable</td>
<td>GPDMA1_CH14</td>
<td>GPDMA1 channel 14 interrupt</td>
<td>0x0000 026C</td>
</tr>
<tr>
<td>140</td>
<td>147</td>
<td>settable</td>
<td>GPDMA1_CH15</td>
<td>GPDMA1 channel 15 interrupt</td>
<td>0x0000 0270</td>
</tr>
<tr>
<td>141</td>
<td>148</td>
<td>settable</td>
<td>HPDMA1_CH8</td>
<td>HPDMA1 channel 8 interrupt</td>
<td>0x0000 0274</td>
</tr>
<tr>
<td>142</td>
<td>149</td>
<td>settable</td>
<td>HPDMA1_CH9</td>
<td>HPDMA1 channel 9 interrupt</td>
<td>0x0000 0278</td>
</tr>
<tr>
<td>143</td>
<td>150</td>
<td>settable</td>
<td>HPDMA1_CH10</td>
<td>HPDMA1 channel 10 interrupt</td>
<td>0x0000 027C</td>
</tr>
<tr>
<td>144</td>
<td>151</td>
<td>settable</td>
<td>HPDMA1_CH11</td>
<td>HPDMA1 channel 11 interrupt</td>
<td>0x0000 0280</td>
</tr>
<tr>
<td>145</td>
<td>152</td>
<td>settable</td>
<td>HPDMA1_CH12</td>
<td>HPDMA1 channel 12 interrupt</td>
<td>0x0000 0284</td>
</tr>
<tr>
<td>146</td>
<td>153</td>
<td>settable</td>
<td>HPDMA1_CH13</td>
<td>HPDMA1 channel 13 interrupt</td>
<td>0x0000 0288</td>
</tr>
<tr>
<td>147</td>
<td>154</td>
<td>settable</td>
<td>HPDMA1_CH14</td>
<td>HPDMA1 channel 14 interrupt</td>
<td>0x0000 028C</td>
</tr>
<tr>
<td>148</td>
<td>155</td>
<td>settable</td>
<td>HPDMA1_CH15</td>
<td>HPDMA1 channel 15 interrupt</td>
<td>0x0000 0290</td>
</tr>
<tr>
<td>149</td>
<td>156</td>
<td>settable</td>
<td>GPU2D</td>
<td>GPU2D global interrupt</td>
<td>0x0000 0294</td>
</tr>
</tbody>
</table>
### Table 142. NVIC(1)

<table>
<thead>
<tr>
<th>NVIC position</th>
<th>Priority</th>
<th>Type of priority</th>
<th>Acronym</th>
<th>Description</th>
<th>Address offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>150</td>
<td>157</td>
<td>settable</td>
<td>GPU2D_ER</td>
<td>GPU2D error interrupt</td>
<td>0x0000 0298</td>
</tr>
<tr>
<td>151</td>
<td>158</td>
<td>settable</td>
<td>TCACHE</td>
<td>GPU cache interrupt</td>
<td>0x0000 029C</td>
</tr>
<tr>
<td>152</td>
<td>159</td>
<td>settable</td>
<td>FDCAN1_IT0</td>
<td>FDCAN1 interrupt 0</td>
<td>0x0000 02A0</td>
</tr>
<tr>
<td>153</td>
<td>160</td>
<td>settable</td>
<td>FDCAN1_IT1</td>
<td>FDCAN1 interrupt 1</td>
<td>0x0000 02A4</td>
</tr>
<tr>
<td>154</td>
<td>161</td>
<td>settable</td>
<td>FDCAN2_IT0</td>
<td>FDCAN2 interrupt 0</td>
<td>0x0000 02A8</td>
</tr>
<tr>
<td>155</td>
<td>162</td>
<td>settable</td>
<td>FDCAN2_IT1</td>
<td>FDCAN2 interrupt 1</td>
<td>0x0000 02AC</td>
</tr>
</tbody>
</table>

When different signals are connected to the same NVIC interrupt line, they are OR-ed.
20 Extended interrupt and event controller (EXTI)

The extended interrupt and event controller (EXTI) manages wakeup through configurable and direct event inputs. It provides wakeup requests to the power control, and generates interrupt requests to the CPU NVIC and events to the CPU event input.

The EXTI wakeup requests allow the system to be woken up from Stop mode, and the CPU to be woken up from CStop mode.

Both the interrupt request and event request generation can also be used in Run modes.

20.1 EXTI main features

The EXTI main features are as follows:

- All event inputs allow the CPU to wakeup and to generate a CPU interrupt and/or CPU event

The asynchronous event inputs are classified in 2 groups:

- Configurable events (signals from I/Os or peripherals able to generate a pulse) that feature:
  - Selectable active trigger edge
  - Interrupt pending status register bit
  - Individual interrupt and event generation mask
  - SW trigger possibility

- Direct events (interrupt and wakeup sources from other peripherals, requiring to be cleared in the peripheral) that feature:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask
  - No SW trigger possibility.

20.2 EXTI block diagram

As shown in Figure 155, the EXTI consists of a register block accessed via an APB interface, an event input Trigger block, and a masking block.

The register block contains all EXTI registers.

The event input trigger block provides event input edge triggering logic.

The masking block provides the event input distribution to the different wakeup, interrupt and event outputs, and their masking.
20.2.1 EXTI connections between peripherals and CPU

The peripherals able to generate wakeup events when the system is in Stop mode or the CPU is in CStop mode are connected to an EXTI configurable event input or direct event input:

- Peripheral signals that generate a pulse are connected to an EXTI configurable event input. For these events the EXTI provides a CPU status pending bit that has to be cleared.
- Peripheral interrupt and wakeup sources that have to be cleared in the peripheral are connected to an EXTI direct event input. There is no CPU status pending bit within the EXTI. The interrupt or wakeup is cleared by the CPU in the peripheral.

The CPU interrupts are connected to their respective CPU NVIC, and, similarly, the CPU event is connected to the CPU rxev input.

The EXTI wakeup signals are connected to the PWR block, and are used to wakeup the CPU.
20.3 EXTI functional description

Depending on the EXTI event input type and wakeup target(s), different logic implementations are used. The applicable features are controlled from register bits:

- Active trigger edge enable, by EXTI rising trigger selection register (EXTI_RTSR1), EXTI rising trigger selection register (EXTI_RTSR2), EXTI interrupt mask register (EXTI_IMR1), and EXTI falling trigger selection register (EXTI_FTSR1), EXTI falling trigger selection register (EXTI_FTSR2), EXTI interrupt mask register (EXTI_IMR1)
- Software trigger, by EXTI software interrupt event register (EXTI_SWIER1), EXTI software interrupt event register (EXTI_SWIER2), EXTI interrupt mask register (EXTI_IMR1)
- CPU interrupt enable, by EXTI interrupt mask register (EXTI_IMR1), EXTI interrupt mask register (EXTI_IMR2), EXTI interrupt mask register (EXTI_IMR3)
- CPU event enable, by EXTI event mask register (EXTI_EMR1), EXTI interrupt mask register (EXTI_IMR2), EXTI event mask register (EXTI_EMR3)

Table 143. EXTI event input configurations and register control(1)

<table>
<thead>
<tr>
<th>Event input type</th>
<th>Wakeup target(s)</th>
<th>Logic implementation</th>
<th>EXTI_RTSR</th>
<th>EXTI_FTSR</th>
<th>EXTI_SWIER</th>
<th>EXTI_IMR</th>
<th>EXTI_EMR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configurable</td>
<td>CPU</td>
<td>Configurable event input, CPU wakeup logic</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Direct</td>
<td>CPU</td>
<td>Direct event input, CPU wakeup logic</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X indicates that functionality is available.
20.3.1 EXTI configurable event input - CPU wakeup

*Figure 20.3.2* is a detailed representation of the logic associated with configurable event inputs which always wake up the CPU.

*Figure 156. Configurable event triggering logic CPU wake up*

The software interrupt event register allows the system to trigger configurable events by software, writing the EXTI software interrupt event register (EXTI_SWIER1), the EXTI software interrupt event register (EXTI_SWIER2), or the EXTI interrupt mask register (EXTI_IMR1) register bit.

The rising edge EXTI rising trigger selection register (EXTI_RTSR1), EXTI rising trigger selection register (EXTI_RTSR2), EXTI interrupt mask register (EXTI_IMR1), and falling edge EXTI falling trigger selection register (EXTI_FTSR1), EXTI falling trigger selection register (EXTI_FTSR2), EXTI interrupt mask register (EXTI_IMR1) selection registers allow the system to enable and select the configurable event active trigger edge or both edges.

The devices feature dedicated interrupt mask registers, namely EXTI interrupt mask register (EXTI_IMR1) and EXTI interrupt mask register (EXTI_IMR2), EXTI interrupt mask register (EXTI_IMR3), and EXTI pending register (EXTI_PR1), EXTI pending register (EXTI_PR2), for configurable event pending request registers. The CPU pending register is only set for an unmasked CPU interrupt. Each event provides a individual CPU interrupt to the CPU NVIC. The configurable event interrupts need to be acknowledged by software in the EXTI_PR register.
The devices feature dedicated event mask registers, i.e. EXTI event mask register (EXTI_EMR1), EXTI interrupt mask register (EXTI_IMR2), and EXTI event mask register (EXTI_EMR3). The enabled event then generates an event on the CPU. All events for a CPU are OR-ed together into a single CPU event signal. The CPU pending register (EXTI_PR) is not set for an unmasked CPU event.

When a CPU interrupt or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the CPU clock is woken up before the asynchronous edge detection circuit is reset.

Note: A detected configurable event, enabled by the CPU, is only cleared when the CPU wakes up.

20.3.2 EXTI direct event input - CPU wakeup

Figure 157 is a detailed representation of the logic associated with direct event inputs waking up the CPU.

Direct events only provide CPU interrupt enable and CPU event enable functionality.

1. The CPU interrupt for asynchronous direct event inputs (peripheral Wakeup signals) is synchronized with the CPU clock. The synchronous direct event inputs (peripheral interrupt signals), after the asynchronous edge detection, are directly sent to the CPU interrupt without resynchronization.
20.4 EXTI event input mapping

For the sixteen GPIO event inputs the associated IPORT pin has to be selected in the SBS register SBS_EXTICRn. The same pin from each IPORT maps to the corresponding EXTI event input.

The wakeup capabilities of each event input are detailed in Table 144. An event input can wake up the CPU.

The EXTI event inputs with a connection to the CPU NVIC are indicated in the Connection to NVIC column. For the EXTI events not having a connection to the NVIC, the peripheral interrupt is directly connected to the NVIC in parallel with the connection to the EXTI.

All EXTI event inputs are OR-ed together and connected to the CPU event input (rxev).

Table 144. EXTI event input mapping

<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 15</td>
<td>EXTI[15:0]</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>16</td>
<td>PVD and AVD (^{(1)})</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>17</td>
<td>RTC alarms,</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>18</td>
<td>RTC tamper, RTC timestamp, LSE_CSS (^{(2)})</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>19</td>
<td>RTC wakeup timer</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>20</td>
<td>VBUS_FS_PLUG_UNPLUG</td>
<td>Configurable</td>
<td>Yes(^{(3)})</td>
</tr>
<tr>
<td>21</td>
<td>VBUS_HS_PLUG_UNPLUG</td>
<td>Configurable</td>
<td>Yes(^{(3)})</td>
</tr>
<tr>
<td>22</td>
<td>I2C1 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>23</td>
<td>I2C2 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>24</td>
<td>I2C3 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>25</td>
<td>I3C1 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>26</td>
<td>USART1 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>27</td>
<td>USART2 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>28</td>
<td>USART3 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>UART4 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>31</td>
<td>UART5 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>32</td>
<td>UART7 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>33</td>
<td>UART8 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>34</td>
<td>ADF wakeup</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>35</td>
<td>LPUART1 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>36</td>
<td>SPI1 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>37</td>
<td>SPI2 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>38</td>
<td>SPI3 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>39</td>
<td>SPI4 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Table 144. EXTI event input mapping (continued)

<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>SPI5 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>41</td>
<td>SPI6 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>42</td>
<td>MDIO wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>43</td>
<td>USB_OTG_FS wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>44</td>
<td>USB_OTG_HS wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>45</td>
<td>UCPD wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>46</td>
<td>ETH_ wakeup(^{(3)})</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>47</td>
<td>LPTIM1 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>48</td>
<td>LPTIM2 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>49</td>
<td>LPTIM2 CH1</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>50</td>
<td>LPTIM3 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>51</td>
<td>LPTIM3 CH1</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>52</td>
<td>LPTIM4 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>53</td>
<td>LPTIM5 wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>54</td>
<td>HDMI-CEC wakeup</td>
<td>Configurable</td>
<td>Yes</td>
</tr>
<tr>
<td>55</td>
<td>WKUP1</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>56</td>
<td>WKUP2</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>57</td>
<td>WKUP3</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>58</td>
<td>WKUP4</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>59</td>
<td>WGLS wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>60</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>61</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>62</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>63</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>64</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>65</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>66</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>67</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>68</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>69</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>70</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>71</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>72</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>73</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>74</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
20.5 **EXTI functional behavior**

The direct event inputs are enabled in the respective peripheral generating the event. The configurable events are enabled by enabling at least one of the trigger edges.

An event only wakes up the CPU when the event-associated CPU interrupt is unmasked and/or the CPU event is unmasked.

<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>76</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>77</td>
<td>DTS wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>78</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>79</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>80</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. PVD and AVD signals are OR-ed together on the same EXTI event input.
2. RTC Tamper, RTC timestamp and LSE_CSS are OR-ed together on the same EXTI event input.
3. lpi_intr_o OR pmt_intr_o

### Table 144. EXTI event input mapping (continued)

<table>
<thead>
<tr>
<th>Event input</th>
<th>Source</th>
<th>Event input type</th>
<th>Connection to NVIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>75</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>DTS wakeup</td>
<td>Direct</td>
<td>Yes</td>
</tr>
<tr>
<td>78</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 145. Masking functionality

<table>
<thead>
<tr>
<th>Interrupt enable MRx bits of EXTI_IMR</th>
<th>Event enable MRx bits of EXTI_EMR</th>
<th>Configurable event inputs PRx bits of EXTI_PR</th>
<th>CPU Interrupt</th>
<th>CPU Event</th>
<th>CPU Wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>No</td>
<td>Masked</td>
<td>Masked</td>
<td>Masked</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>No</td>
<td>Masked</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Status latched</td>
<td>Yes</td>
<td>Masked</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Status latched</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

For configurable event inputs, when the enabled edge(s) occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding pending PRx bit in EXTI_PR is set and the CPU interrupt signal is activated. EXTI_PR PRx pending bit must be cleared by software writing it to ‘1’. This clears the CPU interrupt.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit. When the associated CPU interrupt is unmasked the corresponding CPU interrupt signal is activated.

The CPU event has to be unmasked to generate an event. When the enabled edge(s) occur on the event input a CPU event pulse is generated. There is no CPU event pending bit.

Both a CPU interrupt and a CPU event may be enabled on the same event input. They both trigger the same event input condition(s).

For the configurable event inputs an event input request can be generated by software when writing a ‘1’ in the software interrupt/event register EXTI_SWIER.
20.5.1 EXTI CPU interrupt procedure

- Unmask the event input interrupt by setting the corresponding mask bits in the EXTI_IMR register.
- For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- Enable the associated interrupt source in the CPU NVIC or use the SEVONPEND, so that an interrupt coming from the CPU interrupt signal is detectable by the CPU after a WFI/WFE instruction.
  - For configurable event inputs the associated EXTI pending bit needs to be cleared.

20.5.2 EXTI CPU event procedure

- Unmask the event input by setting the corresponding mask bits of the EXTI_EMR register.
- For configurable event inputs, enable the event input by setting either one or both the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- The CPU event signal is detected by the CPU after a WFE instruction.
  - For configurable event inputs there is no EXTI pending bit to clear.

20.5.3 EXTI CPU wakeup procedure

- Unmask the event input by setting at least one of the corresponding mask bits in the EXTI_IMR and/or EXTI_EMR registers. The CPU wakeup is generated at the same time as the unmasked CPU interrupt and/or CPU event.
- For configurable event inputs, enable the event input by setting either one or both of the corresponding trigger edge enable bits in EXTI_RTSR and EXTI_FTSR registers.
- Direct events automatically generate a CPU wakeup.

20.5.4 EXTI software interrupt/event trigger procedure

Any of the configurable event inputs can be triggered from the software interrupt/event register (the associated CPU interrupt and/or CPU event must be enabled by their respective procedure).

- Enable the event input by setting at least one of the corresponding edge trigger bits in the EXTI_RTSR and/or EXTI_FTSR registers.
- Unmask the software interrupt/event trigger by setting at least one of the corresponding mask bits in the EXTI_IMR and/or EXTI_EMR registers.
- Trigger the software interrupt/event by writing “1” to the corresponding bit in the EXTI_SWIER register.
- The event input may be disabled by clearing the EXTI_RTSR and EXTI_FTSR register bits.

Note: An edge on the configurable event input also triggers an interrupt/event.
20.6 EXTI registers

The registers can only be accessed with a 32-bit (word). A byte or half-word cannot be read or written.

20.6.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT21</td>
<td>RT20</td>
<td>RT19</td>
<td>RT18</td>
<td>RT17</td>
<td>RT16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 21:0 **RT[21:0]**: Rising trigger event configuration bit of configurable event input x.\(^{(1)}\)

0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

20.6.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT21</td>
<td>FT20</td>
<td>FT19</td>
<td>FT18</td>
<td>FT17</td>
<td>FT16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 21:0 **FT[21:0]**: Falling trigger event configuration bit of configurable event input x.\(^{(1)}\)

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set. Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.
### 20.6.3 EXTI software interrupt event register (EXTI_SWIER1)

**Address offset:** 0x08  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 31:22: Reserved, must be kept at reset value.
- Bits 21:0 **SWx**: Software interrupt on event x (x = 21 to 0).  
  - This bitfield always returns 0 when read.  
  - Writing 0 has no effect.  
  - Writing a 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

### 20.6.4 EXTI rising trigger selection register (EXTI_RTCSR2)

**Address offset:** 0x20  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

- Bits 31:23: Reserved, must be kept at reset value.
- Bit 22 **RT54**: Rising trigger event configuration bit of configurable event input x+32.  
  - 0: Rising trigger disabled (for event and interrupt) for input line  
  - 1: Rising trigger enabled (for event and interrupt) for input line  
- Bits 21:20: Reserved, must be kept at reset value.
- Bit 19 **RT51**: Rising trigger event configuration bit of configurable event input x+32.  
  - 0: Rising trigger disabled (for event and interrupt) for input line  
  - 1: Rising trigger enabled (for event and interrupt) for input line  
- Bit 18: Reserved, must be kept at reset value.
- Bit 17 **RT49**: Rising trigger event configuration bit of configurable event input x+32.  
  - 0: Rising trigger disabled (for event and interrupt) for input line  
  - 1: Rising trigger enabled (for event and interrupt) for input line  
- Bits 16:15: Reserved, must be kept at reset value.
Bit 14 **RT46**: Rising trigger event configuration bit of configurable event input x+32.\(^{(1)}\)

0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

Bits 13:3 Reserved, must be kept at reset value.

Bit 2 **RT34**: Rising trigger event configuration bit of configurable event input x+32.\(^{(1)}\)

0: Rising trigger disabled (for event and interrupt) for input line
1: Rising trigger enabled (for event and interrupt) for input line

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

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.

   If a rising edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
   Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

### 20.6.5 EXTI falling trigger selection register ( EXTI_FTSR2 )

Address offset: 0x24

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FT54</td>
<td>FT51</td>
<td>FT49</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 22 **FT54**: Falling trigger event configuration bit of configurable event input x+32.\(^{(1)}\)

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **FT51**: Falling trigger event configuration bit of configurable event input x+32.\(^{(1)}\)

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bit 18 Reserved, must be kept at reset value.

Bit 17 **FT49**: Falling trigger event configuration bit of configurable event input x+32.\(^{(1)}\)

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **FT46**: Falling trigger event configuration bit of configurable event input x+32.\(^{(1)}\)

0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line
20.6.6 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SW46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 13:3 Reserved, must be kept at reset value.

Bit 2 **FT34:** Falling trigger event configuration bit of configurable event input x+32.(1)
0: Falling trigger disabled (for event and interrupt) for input line
1: Falling trigger enabled (for event and interrupt) for input line

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

1. The configurable event inputs are edge triggered, no glitch must be generated on these inputs.
   If a falling edge on the configurable event input occurs during writing of the register, the associated pending bit is not set.
   Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

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

Bit 22 **SW54:** Software interrupt on event x+32
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **SW51:** Software interrupt on event x+32
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **SW49:** Software interrupt on event x+32
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **SW46:** Software interrupt on event x+32
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line x. This bit is auto cleared by HW.
Bits 13:3  Reserved, must be kept at reset value.

Bit 2  **SW34**: Software interrupt on event x+32
Always returns 0 when read.
0: Writing 0 has no effect.
1: Writing a 1 to this bit triggers an event on line x. This bit is auto cleared by HW.

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

### 20.6.7  EXTI interrupt mask register (EXTI_IMR1)

Address offset: 0x80  
Reset value: 0xFFC0 0000

<table>
<thead>
<tr>
<th>IM31</th>
<th>IM30</th>
<th>IM29</th>
<th>IM28</th>
<th>IM27</th>
<th>IM26</th>
<th>IM25</th>
<th>IM24</th>
<th>IM23</th>
<th>IM22</th>
<th>IM21</th>
<th>IM20</th>
<th>IM19</th>
<th>IM18</th>
<th>IM17</th>
<th>IM16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>IM15</td>
<td>IM14</td>
<td>IM13</td>
<td>IM12</td>
<td>IM11</td>
<td>IM10</td>
<td>IM9</td>
<td>IM8</td>
<td>IM7</td>
<td>IM6</td>
<td>IM5</td>
<td>IM4</td>
<td>IM3</td>
<td>IM2</td>
<td>IM1</td>
<td>IM0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:22  **IM[31:22]**: CPU interrupt mask on direct event input x\(^{(1)}\)
0: Interrupt request from line x is masked
1: Interrupt request from line x is unmasked

Bits 21:0  **IM[21:0]**: CPU interrupt mask on configurable event input x \(^{(2)}\)
0: Interrupt request from line x is masked
1: Interrupt request from line x is unmasked

1. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.
2. The reset value for configurable event inputs is set to ‘0’ in order to disable the interrupt by default.
## 20.6.8 EXTI event mask register (EXTI_EMR1)

Address offset: 0x84  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **EM[31:0]**: CPU event mask on event input x  
0: Event request from line x is masked  
1: Event request from line x is unmasked

## 20.6.9 EXTI pending register (EXTI_PR1)

Address offset: 0x88  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR31</td>
<td>PR30</td>
<td>PR29</td>
<td>PR28</td>
<td>PR27</td>
<td>PR26</td>
<td>PR25</td>
<td>PR24</td>
<td>PR23</td>
<td>PR22</td>
<td>PR21</td>
<td>PR20</td>
<td>PR19</td>
<td>PR18</td>
<td>PR17</td>
<td>PR16</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PR15</td>
<td>PR14</td>
<td>PR13</td>
<td>PR12</td>
<td>PR11</td>
<td>PR10</td>
<td>PR9</td>
<td>PR8</td>
<td>PR7</td>
<td>PR6</td>
<td>PR5</td>
<td>PR4</td>
<td>PR3</td>
<td>PR2</td>
<td>PR1</td>
<td>PR0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

Bits 31:22 Reserved, must be kept at reset value.  
Bits 21:0 **PR[21:0]**: Configurable event inputs x Pending bit  
0: No trigger request occurred  
1: selected trigger request occurred  
This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.
20.6.10 EXTI interrupt mask register (EXTI_IMR2)

Address offset: 0x90
Reset value: 0xFFF5 FFFF

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **IM[59:32]**: CPU interrupt mask on direct event input x+32 (x = 59 to 32)
0: Interrupt request from line x is masked
1: Interrupt request from line x is unmasked

20.6.11 EXTI event mask register (EXTI_EMR2)

Address offset: 0x94
Reset value: 0x0000 0000

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:0 **EM[59:32]**: CPU event mask on event input x+32 (x = 59 to 32)
0: Event request from line x is masked
1: Event request from line x is unmasked
## 20.6.12 EXTI pending register (EXTI_PR2)

Address offset: 0x98
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PR46</td>
<td>PR45</td>
<td>PR44</td>
<td>PR43</td>
<td>PR42</td>
<td>PR41</td>
<td>PR40</td>
<td>PR39</td>
<td>PR38</td>
<td>PR37</td>
<td>PR36</td>
<td>PR35</td>
<td>PR34</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 22 **PR54**: Configurable event inputs x+32 Pending bit
- 0: No trigger request occurred
- 1: selected trigger request occurred
  This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 21:20: Reserved, must be kept at reset value.

Bit 19 **PR51**: Configurable event inputs x+32 Pending bit
- 0: No trigger request occurred
- 1: selected trigger request occurred
  This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bit 18: Reserved, must be kept at reset value.

Bit 17 **PR49**: Configurable event inputs x+32 Pending bit
- 0: No trigger request occurred
- 1: selected trigger request occurred
  This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 16:15: Reserved, must be kept at reset value.

Bit 14 **PR46**: Configurable event inputs x+32 Pending bit
- 0: No trigger request occurred
- 1: selected trigger request occurred
  This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 13:3: Reserved, must be kept at reset value.

Bit 2 **PR34**: Configurable event inputs x+32 Pending bit
- 0: No trigger request occurred
- 1: selected trigger request occurred
  This bit is set when the selected edge event arrives on the external interrupt line. This bit is cleared by writing a 1 into the bit or by changing the sensitivity of the edge detector.

Bits 1:0: Reserved, must be kept at reset value.
## 20.6.13 EXTI interrupt mask register (EXTI_IMR3)

Address offset: 0xA0
Reset value: 0x0F8B FFFF

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>IM77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **IM[77]**: CPU interrupt mask on direct event input x+64 (1)

0: Interrupt request from line x is masked
1: Interrupt request from line x is unmasked

Bits 12:0 Reserved, must be kept at reset value.

1. The reset value for direct event inputs is set to ‘1’ in order to enable the interrupt by default.

## 20.6.14 EXTI event mask register (EXTI_EMR3)

Address offset: 0xA4
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EM77</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **EM[77]**: CPU event mask on event input x+64

0: Event request from line x is masked
1: Event request from line x is unmasked

Bits 12:0 Reserved, must be kept at reset value.
### 20.6.15 EXTI register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name reset value</th>
<th>0x00</th>
<th>0x04</th>
<th>0x08</th>
<th>0x12-0x1C</th>
<th>0x20</th>
<th>0x24</th>
<th>0x28</th>
<th>0x2C-0x7C</th>
<th>0x80</th>
<th>0x84</th>
<th>0x88</th>
<th>0x90</th>
<th>0x94</th>
<th>0x98</th>
<th>0x99-0x9F</th>
<th>0x100-0x5F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00</td>
<td>EXTI_RTSR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>EXTI_FTSR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>EXTI_SWIER1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x12-0x1C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>EXTI_RTSR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>EXTI_FTSR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>EXTI_SWIER2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C-0x7C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>EXTI_IMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x84</td>
<td>EXTI_EMRI1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>EXTI_PR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x90</td>
<td>EXTI_IMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x94</td>
<td>EXTI_EMRI2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x98</td>
<td>EXTI_PR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x99-0x9F</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x9F-0x100</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 149 for the register boundary addresses.
21 Cyclic redundancy check calculation unit (CRC)

21.1 Introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

21.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-, 16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility option on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes
21.3 CRC functional description

21.3.1 CRC block diagram

**Figure 158. CRC calculation unit block diagram**

21.3.2 CRC internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
</tbody>
</table>

21.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on data width:
- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.
The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register.

For example, 0x1A2B3C4D input data are used for CRC calculation as:
- 0x58D43CB2 with bit-reversal done by byte
- 0xD458B23C with bit-reversal done by half-word
- 0xB23CD458 with bit-reversal done on the full word

The output data can also be reversed by setting the REV_OUT bit in the CRC_CR register.

The operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC calculation. It is not affected by the RESET bit in the CRC_CR register.

**Polynomial programmability**

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7, 8, 16 or 32 bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

*Note: The type of an even polynomial is \( X + X^2 + \ldots + X^n \), while the type of an odd polynomial is \( 1 + X + X^2 + \ldots + X^n \).*

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.
### 21.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes. For the other registers only 32-bit accesses are allowed.

#### 21.4.1 CRC data register (CRC_DR)

**Address offset:** 0x00  
**Reset value:** 0xFFFF FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0** **DR[31:0]:** Data register bits  
This register is used to write new data to the CRC calculator.  
It holds the previous CRC calculation result when it is read.  
If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

#### 21.4.2 CRC independent data register (CRC_IDR)

**Address offset:** 0x04  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0** **IDR[31:0]:** General-purpose 32-bit data register bits  
These bits can be used as a temporary storage location for four bytes.  
This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register.
### 21.4.3 CRC control register (CRC_CR)

Address offset: 0x08  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
</tr>
</tbody>
</table>

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

Bit 7  **REV_OUT**: Reverse output data  
This bit controls the reversal of the bit order of the output data.  
0: Bit order not affected  
1: Bit-reversed output format

Bits 6:5  **REV_IN[1:0]**: Reverse input data  
This bitfield controls the reversal of the bit order of the input data  
00: Bit order not affected  
01: Bit reversal done by byte  
10: Bit reversal done by half-word  
11: Bit reversal done by word

Bits 4:3  **POLYSIZE[1:0]**: Polynomial size  
These bits control the size of the polynomial.  
00: 32 bit polynomial  
01: 16 bit polynomial  
10: 8 bit polynomial  
11: 7 bit polynomial

Bits 2:1  Reserved, must be kept at reset value.

Bit 0  **RESET**: RESET bit  
This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.
21.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10
Reset value: 0xFFFF FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 CRC_INIT[31:0]: Programmable initial CRC value
This register is used to write the CRC initial value.

21.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14
Reset value: 0x04C1 1DB7

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 POL[31:0]: Programmable polynomial
This register is used to write the coefficients of the polynomial to be used for CRC calculation.
If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.
## 21.4.6 CRC register map

### Table 148. CRC register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRC_DR</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>CRC_IDR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x06</td>
<td>CRC_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>CRC_INIT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>CRC_POL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
22 CORDIC coprocessor (CORDIC)

22.1 CORDIC introduction

The CORDIC coprocessor provides hardware acceleration of mathematical functions (mainly trigonometric ones) commonly used in motor control, metering, signal processing, and many other applications.

It speeds up the calculation of these functions compared to a software implementation, making possible the use of a lower operating frequency, or freeing up processor cycles to perform other tasks.

22.2 CORDIC main features

- 24-bit CORDIC rotation engine
- Circular and Hyperbolic modes
- Rotation and Vectoring modes
- Functions: sine, cosine, sinh, cosh, atan, atan2, atanh, modulus, square root, natural logarithm
- Programmable precision
- Low latency AHB slave interface
- Results can be read as soon as ready, without polling or interrupt
- DMA read and write channels
- Multiple register read/write by DMA

22.3 CORDIC functional description

22.3.1 General description

The CORDIC is a cost-efficient successive approximation algorithm for evaluating trigonometric and hyperbolic functions.

In trigonometric (circular) mode, the sine and cosine of an angle $\theta$ are determined by rotating the unit vector [1, 0] through decreasing angles until the cumulative sum of the rotation angles equals the input angle $\theta$. The x and y cartesian components of the rotated vector then correspond, respectively, to the cosine and sine of $\theta$. Inversely, the angle of a vector [x, y] corresponding to arctangent (y / x), is determined by rotating [x, y] through successively decreasing angles to obtain the unit vector [1, 0]. The cumulative sum of the rotation angles gives the angle of the original vector.

The CORDIC algorithm can also be used for calculating hyperbolic functions (sinh, cosh, atanh), by replacing the successive circular rotations by steps along a hyperbole.

Other functions can be derived from the basic functions described above.

22.3.2 CORDIC functions

The first step when using the coprocessor is to select the required function, by programming the FUNC field of the CORDIC_CR register accordingly.
Table 149 lists the functions supported by the CORDIC coprocessor.

### Table 149. CORDIC functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Primary argument (ARG1)</th>
<th>Secondary argument (ARG2)</th>
<th>Primary result (RES1)</th>
<th>Secondary result (RES2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine</td>
<td>angle ( \theta )</td>
<td>modulus ( m )</td>
<td>( m \cdot \cos \theta )</td>
<td>( m \cdot \sin \theta )</td>
</tr>
<tr>
<td>Sine</td>
<td>angle ( \theta )</td>
<td>modulus ( m )</td>
<td>( m \cdot \sin \theta )</td>
<td>( m \cdot \cos \theta )</td>
</tr>
<tr>
<td>Phase</td>
<td>( x )</td>
<td>( y )</td>
<td>( \text{atan2}(y,x) )</td>
<td>( \sqrt{x^2 + y^2} )</td>
</tr>
<tr>
<td>Modulus</td>
<td>( x )</td>
<td>( y )</td>
<td>( \sqrt{x^2 + y^2} )</td>
<td>( \text{atan2}(y,x) )</td>
</tr>
<tr>
<td>Arctangent</td>
<td>( x )</td>
<td>none</td>
<td>( \tan^{-1} x )</td>
<td>none</td>
</tr>
<tr>
<td>Hyperbolic cosine</td>
<td>( x )</td>
<td>none</td>
<td>( \cosh x )</td>
<td>( \sinh x )</td>
</tr>
<tr>
<td>Hyperbolic sine</td>
<td>( x )</td>
<td>none</td>
<td>( \sinh x )</td>
<td>( \cosh x )</td>
</tr>
<tr>
<td>Hyperbolic arctangent</td>
<td>( x )</td>
<td>none</td>
<td>( \tanh^{-1} x )</td>
<td>none</td>
</tr>
<tr>
<td>Natural logarithm</td>
<td>( x )</td>
<td>none</td>
<td>( \ln x )</td>
<td>none</td>
</tr>
<tr>
<td>Square root</td>
<td>( x )</td>
<td>none</td>
<td>( \sqrt{x} )</td>
<td>none</td>
</tr>
</tbody>
</table>

Several functions take two input arguments (ARG1 and ARG2) and some generate two results (RES1 and RES2) simultaneously. This is a side-effect of the algorithm and means that only one operation is needed to obtain two values. This is the case, for example, when performing polar-to-rectangular conversion: \( \sin \theta \) also generates \( \cos \theta \), \( \cos \theta \) also generates \( \sin \theta \). Similarly, for rectangular-to-polar conversion (\( \text{phase}(x,y) \), \( \text{modulus}(x,y) \) and for hyperbolic functions (\( \cosh \theta \), \( \sinh \theta \)).

**Note:** The exponential function, \( \exp x \), can be obtained as the sum of \( \sinh x \) and \( \cosh x \). Furthermore, base \( N \) logarithms, \( \log_N x \), can be derived by multiplying \( \ln x \) by a constant \( K \), where \( K = 1/\ln N \).

For certain functions (atan, log, sqrt) a scaling factor (see Section 22.3.4) can be applied to extend the range of the function beyond the maximum [-1, 1] supported by the q1.31 fixed point format. The scaling factor must be set to 0 for all other circular functions, and to 1 for hyperbolic functions.

### Cosine

#### Table 150. Cosine parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>Angle ( \theta ) in radians, divided by ( \pi )</td>
<td>[-1, 1]</td>
</tr>
<tr>
<td>ARG2</td>
<td>Modulus ( m )</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>RES1</td>
<td>( m \cdot \cos \theta )</td>
<td>[-1, 1]</td>
</tr>
</tbody>
</table>
This function calculates the cosine of an angle in the range \(-\pi\) to \(\pi\). It can also be used to perform polar to rectangular conversion.

The primary argument is the angle \(\theta\) in radians. It must be divided by \(\pi\) before programming ARG1.

The secondary argument is the modulus \(m\). If \(m\) is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the sine of the angle, multiplied by the modulus.

### Sine

This function calculates the sine of an angle in the range \(-\pi\) to \(\pi\). It can also be used to perform polar to rectangular conversion.

The primary argument is the angle \(\theta\) in radians. It must be divided by \(\pi\) before programming ARG1.

The secondary argument is the modulus \(m\). If \(m\) is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the sine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.

### Phase

This function calculates the phase angle \(\theta\) in radians, divided by \(\pi\). It can also be used to perform polar to rectangular conversion.

The primary argument is the angle \(\theta\) in radians. It must be divided by \(\pi\) before programming ARG1.

The secondary argument is the modulus \(m\). If \(m\) is greater than 1, a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the cosine of the angle, multiplied by the modulus.

The secondary result, RES2, is the cosine of the angle, multiplied by the modulus.
This function calculates the phase angle in the range \(-\pi\) to \(\pi\) of a vector \(v = [x, y]\) (also known as \(\text{atan2}(y, x)\)). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If \(|x| > 1\), a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If \(|y| > 1\), a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the phase angle \(\theta\) of the vector \(v\). RES1 must be multiplied by \(\pi\) to obtain the angle in radians. Note that values close to \(\pi\) may sometimes wrap to \(-\pi\) due to the circular nature of the phase angle.

The secondary result, RES2, is the modulus, given by: \(|v| = \sqrt{x^2 + y^2}\). If \(|v| > 1\) the result in RES2 is saturated to 1.

**Modulus**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>x coordinate</td>
<td>[-1, 1]</td>
</tr>
<tr>
<td>ARG2</td>
<td>y coordinate</td>
<td>[-1, 1]</td>
</tr>
<tr>
<td>RES1</td>
<td>Modulus m</td>
<td>[0, 1]</td>
</tr>
<tr>
<td>RES2</td>
<td>Phase angle (\theta)</td>
<td>[-1, 1]</td>
</tr>
<tr>
<td>SCALE</td>
<td>Not applicable</td>
<td>0</td>
</tr>
</tbody>
</table>

This function calculates the magnitude, or modulus, of a vector \(v = [x, y]\). It can also be used to perform rectangular to polar conversion.

The primary argument is the x coordinate, that is, the magnitude of the vector in the direction of the x axis. If \(|x| > 1\), a scaling must be applied in software to adapt it to the q1.31 range of ARG1.

The secondary argument is the y coordinate, that is, the magnitude of the vector in the direction of the y axis. If \(|y| > 1\), a scaling must be applied in software to adapt it to the q1.31 range of ARG2.

The primary result, RES1, is the modulus, given by: \(|v| = \sqrt{x^2 + y^2}\). If \(|v| > 1\) the result in RES1 is saturated to 1.

The secondary result, RES2, is the phase angle \(\theta\) of the vector \(v\). RES2 must be multiplied by \(\pi\) to obtain the angle in radians. Note that values close to \(\pi\) may sometimes wrap to \(-\pi\) due to the circular nature of the phase angle.
Arctangent

Table 154. Arctangent parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>[-1, 1]</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{-n} \cdot \tan^{-1} x ), in radians, divided by ( \pi )</td>
<td>[-1, 1]</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>[0, 7]</td>
</tr>
</tbody>
</table>

This function calculates the arctangent, or inverse tangent, of the input argument \( x \).

The primary argument, ARG1, is the input value, \( x = \tan \theta \). If \(|x| > 1\), a scaling factor of \( 2^{-n} \) must be applied in software such that \(-1 < x \cdot 2^{-n} < 1\). The scaled value \( x \cdot 2^{-n} \) is programmed in ARG1 and the scale factor \( n \) must be programmed in the SCALE parameter.

Note that the maximum input value allowed is \( \tan \theta = 128 \), which corresponds to an angle \( \theta = 89.55 \) degrees. For \(|x| > 128\), a software method must be used to find \( \tan^{-1} x \).

The secondary argument, ARG2, is unused.

The primary result, RES1, is the angle \( \theta = \tan^{-1} x \). RES1 must be multiplied by \( 2^n \cdot \pi \) to obtain the angle in radians.

The secondary result, RES2, is unused.

Hyperbolic cosine

Table 155. Hyperbolic cosine parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>[-0.559, 0.559]</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{-n} \cdot \cosh x )</td>
<td>[0.5, 0.846]</td>
</tr>
<tr>
<td>RES2</td>
<td>( 2^{-n} \cdot \sinh x )</td>
<td>[-0.683, 0.683]</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>1</td>
</tr>
</tbody>
</table>

This function calculates the hyperbolic cosine of a hyperbolic angle \( x \). It can also be used to calculate the exponential functions \( e^x = \cosh x + \sinh x \), and \( e^{-x} = \cosh x - \sinh x \).

The primary argument is the hyperbolic angle \( x \). Only values of \( x \) in the range \([-1.118, 1.118]\) are supported. Since the minimum value of \( \cosh x \) is 1, which is beyond the range of the q1.31 format, a scaling factor of \( 2^{-n} \) must be applied in software. The factor \( n = 1 \) must be programmed in the SCALE parameter.

The secondary argument is not used.
The primary result, RES1, is the hyperbolic cosine, cosh x. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic sine, sinh x. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic sine

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>$x \cdot 2^{-n}$</td>
<td>[-0.559, 0.559]</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>$2^{-n} \cdot \sinh x$</td>
<td>[-0.683, 0.683]</td>
</tr>
<tr>
<td>RES2</td>
<td>$2^{-n} \cdot \cosh x$</td>
<td>[0.5, 0.846]</td>
</tr>
<tr>
<td>SCALE</td>
<td>$n$</td>
<td>1</td>
</tr>
</tbody>
</table>

This function calculates the hyperbolic sine of a hyperbolic angle $x$. It can also be used to calculate the exponential functions $e^x = \cosh x + \sinh x$, and $e^{-x} = \cosh x - \sinh x$.

The primary argument is the hyperbolic angle $x$. Only values of $x$ in the range $-1.118$ to $+1.118$ are supported. For all input values, a scaling factor of $2^{-n}$ must be applied in software, where $n = 1$. The scaled value $x \cdot 0.5$ is programmed in ARG1 and the factor $n = 1$ must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result, RES1, is the hyperbolic sine, sinh $x$. RES1 must be multiplied by 2 to obtain the correct result.

The secondary result, RES2, is the hyperbolic cosine, cosh $x$. RES2 must be multiplied by 2 to obtain the correct result.

Hyperbolic arctangent

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>$x \cdot 2^{-n}$</td>
<td>[-0.403, 0.403]</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>$2^{-n} \cdot \tanh x$</td>
<td>[-0.559, 0.559]</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>$n$</td>
<td>1</td>
</tr>
</tbody>
</table>

This function calculates the hyperbolic arctangent of the input argument $x$.

The primary argument is the input value $x$. Only values of $x$ in the range $-0.806$ to $+0.806$ are supported. The value $x$ must be scaled by a factor $2^{-n}$, where $n = 1$. The scaled value
x \cdot 0.5 is programmed in ARG1 and the factor n = 1 must be programmed in the SCALE parameter.

The secondary argument is not used.

The primary result is the hyperbolic arctangent, \( \text{atanh} \ x \). RES1 must be multiplied by 2 to obtain the correct value.

The secondary result is not used.

**Natural logarithm**

**Table 158. Natural logarithm parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^n )</td>
<td>([0.054 \ 0.875])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{(n+1)} \ln x )</td>
<td>([-0.279 \ 0.137])</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>n</td>
<td>([1 \ 4])</td>
</tr>
</tbody>
</table>

This function calculates the natural logarithm of the input argument \( x \).

The primary argument is the input value \( x \). Only values of \( x \) in the range 0.107 to 9.35 are supported. The value \( x \) must be scaled by a factor \( 2^{-n} \), such that \( x \cdot 2^{-n} < 1-2^{-n} \). The scaled value \( x \cdot 2^{-n} \) is programmed in ARG1 and the factor \( n \) must be programmed in the SCALE parameter.

**Table 159** lists the valid scaling factors, \( n \), and the corresponding ranges of \( x \) and ARG1.

**Table 159. Natural log scaling factors and corresponding ranges**

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x ) range</th>
<th>ARG1 range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 0.107 \leq x &lt; 1 )</td>
<td>( 0.0535 \leq \text{ARG1} &lt; 0.5 )</td>
</tr>
<tr>
<td>2</td>
<td>( 1 \leq x &lt; 3 )</td>
<td>( 0.25 \leq \text{ARG1} &lt; 0.75 )</td>
</tr>
<tr>
<td>3</td>
<td>( 3 \leq x &lt; 7 )</td>
<td>( 0.375 \leq \text{ARG1} &lt; 0.875 )</td>
</tr>
<tr>
<td>4</td>
<td>( 7 \leq x \leq 9.35 )</td>
<td>( 0.4375 \leq \text{ARG1} &lt; 0.584 )</td>
</tr>
</tbody>
</table>

The secondary argument is not used.

The primary result is the natural logarithm, \( \ln x \). RES1 must be multiplied by \( 2^{(n+1)} \) to obtain the correct value.

The secondary result is not used.
Square root

This function calculates the square root of the input argument \( x \).

The primary argument is the input value \( x \). Only values of \( x \) in the range 0.027 to 2.34 are supported. The value \( x \) must be scaled by a factor \( 2^{-n} \), such that \( x \cdot 2^{-n} < (1 - 2^{(n-2)}) \).

The scaled value \( x \cdot 2^{-n} \) is programmed in ARG1 and the factor \( n \) must be programmed in the SCALE parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARG1</td>
<td>( x \cdot 2^{-n} )</td>
<td>([0.027 \ 0.875])</td>
</tr>
<tr>
<td>ARG2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>RES1</td>
<td>( 2^{-n} \sqrt{x} )</td>
<td>([0.04 \ 1])</td>
</tr>
<tr>
<td>RES2</td>
<td>Not applicable</td>
<td>-</td>
</tr>
<tr>
<td>SCALE</td>
<td>( n )</td>
<td>([0 \ 2])</td>
</tr>
</tbody>
</table>

Table 160. Square root parameters

Table 161 lists the valid scaling factors, \( n \), and the corresponding ranges of \( x \) and ARG1.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x ) range</th>
<th>ARG1 range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.027 ( \leq x &lt; 0.75 )</td>
<td>0.027 ( \leq ) ARG1 ( &lt; 0.75 )</td>
</tr>
<tr>
<td>1</td>
<td>0.75 ( \leq x &lt; 1.75 )</td>
<td>0.375 ( \leq ) ARG1 ( &lt; 0.875 )</td>
</tr>
<tr>
<td>2</td>
<td>1.75 ( \leq x \leq 2.341 )</td>
<td>0.4375 ( \leq ) ARG1 ( \leq 0.585 )</td>
</tr>
</tbody>
</table>

The secondary argument is not used.

The primary result is the square root of \( x \). RES1 must be multiplied by \( 2^n \) to obtain the correct value.

The secondary result is not used.

22.3.3 Fixed point representation

The CORDIC operates in fixed point signed integer format. Input and output values can be either q1.31 or q1.15.

In q1.31 format, numbers are represented by one sign bit and 31 fractional bits (binary decimal places). The numeric range is therefore -1 (0x80000000) to 1 - \( 2^{-31} \) (0x7FFFFFFF).

In q1.15 format, the numeric range is 1 (0x8000) to 1 - \( 2^{-15} \) (0x7FFF). This format has the advantage that two input arguments can be packed into a single 32-bit write, and two results can be fetched in one 32-bit read.

22.3.4 Scaling factor

Several of the functions listed in Section 22.3.2 specify a scaling factor, SCALE. This allows the function input range to be extended to cover the full range of values supported by the CORDIC, without saturating the input, output, or internal registers. If the scaling factor is
required, it must be calculated by software and programmed into the SCALE field of the CORDIC_CSR register. The input arguments must be scaled accordingly before programming the scaled values in the CORDIC_WDATA register. The scaling must also be undone on the results read from the CORDIC_RDATA register.

Note: The scaling factor entails a loss of precision due to truncation of the scaled value.

22.3.5 Precision

The precision of the result is dependent on the number of CORDIC iterations. The algorithm converges at a constant rate of one binary digit per iteration for trigonometric functions (sine, cosine, phase, modulus), see Figure 159.

For hyperbolic functions (hyperbolic sine, hyperbolic cosine, natural logarithm), the convergence rate is less constant due to the peculiarities of the CORDIC algorithm (see Figure 160). The square root function converges at roughly twice the speed of the hyperbolic functions (see Figure 161).

Figure 159. CORDIC convergence for trigonometric functions
Figure 160. CORDIC convergence for hyperbolic functions

Hyperbolic Sine/Cosine Convergence of q1.23 fixed point Cordic

Max absolute error

q1.15

q1.31

Iterations
Figure 161. CORDIC convergence for square root

Note: The convergence rate decreases as the quantization error starts to become significant.

The CORDIC can perform four iterations per clock cycle. For each function, the maximum error remaining after every four iterations is shown in Table 162, together with the number of clock cycles required to reach that precision. From this table, the desired number of cycles can be determined and programmed in the PRECISION field of the CORDIC_CR register. The coprocessor stops as soon as the programmed number of iterations is completed, and the result can be read immediately.

Table 162. Precision vs. number of iterations

<table>
<thead>
<tr>
<th>Function</th>
<th>Number of iterations</th>
<th>Number of cycles</th>
<th>Max residual error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>q1.31 format</td>
</tr>
<tr>
<td>Sin, Cos, Phase(2), Mod, Atan(4)</td>
<td>4</td>
<td>1</td>
<td>$2^{-3}$</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2</td>
<td>$2^{-7}$</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>3</td>
<td>$2^{-11}$</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>4</td>
<td>$2^{-15}$</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>5</td>
<td>$2^{-18}$</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>6</td>
<td>$2^{-19}$</td>
</tr>
</tbody>
</table>
The fastest way to use the coprocessor is to preprogram the CORDIC_CSR register with the function to be performed (FUNC), the desired number of clock cycles (PRECISION), the size of the input and output values (ARGSIZE, RESSIZE), the number of input arguments (NARGS) and/or results (NRES), and the scaling factor (SCALE), if applicable.

The calculation is triggered by writing the input arguments to the CORDIC_WDATA register. As soon as the correct number of input arguments has been written (and any ongoing calculation has finished), a new calculation is launched using these input arguments and the current CORDIC_CSR settings. There is no need to reprogram the CORDIC_CSR register if there is no change.

If a dual 32-bit input argument is needed (ARGSIZE = 0, NARGS = 1), the primary input argument (ARG1) must be written first, followed by the secondary argument (ARG2). If the secondary argument remains unchanged for a series of calculations, the second write can be avoided, by reprogramming the number of arguments to one (NARGS = 0), once the first calculation has started. The secondary argument retains its programmed value as long as the function is not changed.

Note: ARG2 is set to +1 (0x7FFFFFFF) after a reset.

If two 16-bit arguments are used (ARGSIZE = 1) they must be packed into a 32-bit word, with ARG1 in the least significant half-word and ARG2 in the most significant half-word. The packed 32-bit word is then written to the CORDIC_WDATA register. Only one write is needed in this case (NARGS = 0).

For functions taking only one input argument, ARG1, it is recommended to set NARGS = 0. If NARGS = 1, a second write to CORDIC_WDATA must be performed to trigger the calculation. The ARG2 data in this case is not used.
Once the calculation starts, any attempt to read the CORDIC_RDATA register inserts bus wait-states until the calculation is completed, before returning the result. It is then possible for the software to write the input and immediately read the result without polling to see if it is valid. Alternatively, the processor can wait for the appropriate number of clock cycles before reading the result. This time can be used to program the CORDIC_CSR register for the next calculation, and prepare the next input data, if needed. The CORDIC_CSR register can be reprogrammed while a calculation is in progress, without affecting the result of the ongoing calculation. In the same way, the CORDIC_WDATA register can be updated with the next argument(s) once the previous ones have been taken into account. The next arguments and settings remain pending until the previous calculation has completed.

When a calculation is finished, the result(s) can be read from the CORDIC_RDATA register. If two 32-bit results are expected (NRES = 1, RESSIZE = 0), the primary result (RES1) is read out first, followed by the secondary result (RES2). If only one 32-bit result is expected (NRES = 0, RESSIZE = 0), then RES1 is output on the first read.

If 16-bit results are expected (RESSIZE = 1), a single read to CORDIC_RDATA fetches both results packed into a 32-bit word. RES1 is in the lower half-word, and RES2 in the upper half-word. In this case, it is recommended to program NRES = 0. If NRES = 1, a second read of CORDIC_RDATA must be performed to free up the CORDIC for the next operation. The data from this second read must be discarded.

The next calculation starts when the expected number of results has been read, provided the expected number of arguments has been written. This means that at any time, there can be a calculation in progress, or waiting for the results to be read, and an operation pending. Any further access to CORDIC_WDATA while an operation is pending cancels it and overwrites the data.

The following sequence summarizes the use of the CORDIC_IP in zero-overhead mode:
1. Program the CORDIC_CSR register with the appropriate settings
2. Program the argument(s) for the first calculation in the CORDIC_WDATA register. This launches the first calculation.
3. If needed, update the CORDIC_CSR register settings for the next calculation.
4. Program the argument(s) for the next calculation in the CORDIC_WDATA register.
5. Read the result(s) from the CORDIC_RDATA register. This triggers the next calculation.
6. Go to step 3.

### 22.3.7 Polling mode

When a new result is available in the CORDIC_RDATA register, the RRDY flag is set in the CORDIC_CSR register. The flag can be polled by reading the register. It is reset by reading the CORDIC_RDATA register (once or twice, depending on the NRES field of the CORDIC_CSR register).

Polling the RRDY flag takes slightly longer than reading the CORDIC_RDATA register directly, since the result is not read as soon as it is available. The processor and bus interface are not stalled while reading the CORDIC_CSR register, so this mode may be of interest if stalling the processor is not acceptable (for example, if low latency interrupts must be serviced).
22.3.8 **Interrupt mode**

By setting the interrupt enable (IE) bit in the CORDIC_CSR register, an interrupt is generated whenever the RRDY flag is set. The interrupt is cleared when the flag is reset.

This mode allows the result of the calculation to be read under interrupt service routine, and hence given a priority relative to other tasks. However, it is slower than directly reading the result, or polling the flag, due to the interrupt handling delays.

22.3.9 **DMA mode**

If the DMA write enable (DMAWEN) bit is set in the CORDIC_CSR register, and no operation is pending, a DMA write channel request is generated. The DMA controller can transfer a primary input argument (ARG1) from memory into the CORDIC_WDATA register. Writing into the register deasserts the DMA request. If NARGS = 1 in the CORDIC_CSR register, a second DMA write channel request is generated to transfer the secondary input argument (ARG2) into the CORDIC_WDATA register. When all input arguments have been written, and any ongoing calculation has been completed (by reading the results), a new calculation is started and another DMA write channel request is generated.

If the DMA read enable (DMAREN) bit is set in the CORDIC_CSR register, the RRDY flag going active generates a DMA read channel request. The DMA controller can then transfer the primary result (RES1) from the CORDIC_RDATA register to memory. Reading the register deasserts the DMA request. If NRES = 1 in the CORDIC_CSR register, a second DMA request is generated to read out the secondary result (RES2). When all results have been read, the RRDY flag is deasserted.

The DMA read and write channels can be enabled separately. If both channels are enabled, the CORDIC can autonomously perform repeated calculations on a buffer of data without processor intervention. This allows the processor to perform other tasks. The DMA controller is operating in memory-to-peripheral mode for the write channel, and peripheral-to-memory mode for the read channel. The sequence is started by the processor setting the DMAWEN flag, the DMA read and write requests are generated as fast as the CORDIC can process the data.

In some cases, the input data may be stored in memory, and the output is transferred at regular intervals to another peripheral, such as a digital-to-analog converter. In this case, the destination peripheral generates a DMA request each time it needs a new data. The DMA controller can directly fetch the next sample from the CORDIC_RDATA register (in this case the DMA controller is operating in memory-to-peripheral mode, even though the source is a peripheral register). The act of reading the result allows the CORDIC to start a new calculation, which in turn generates a DMA write channel request, and the DMA controller transfers the next input value to the CORDIC_WDATA register. The DMA write channel is enabled (DMAWEN = 1), but the read channel must not be enabled.

In a similar way, data coming from another peripheral, such as an ADC, can be transferred directly to the CORDIC_WDATA register (in peripheral-to-memory mode). The DMA write channel must not be enabled. The CORDIC processes the input data and generates a DMA read request when complete, if DMAREN = 1. The DMA controller then transfers the result from CORDIC_RDATA register to memory (peripheral-to-memory mode).

**Note:** No DMA request is generated to program the CORDIC_CSR register. DMA mode is therefore useful only when repeatedly performing the same function with the same settings. The scale factor cannot be changed during a series of DMA transfers.
Note: Each DMA request must be acknowledged, as a result of the DMA performing an access to the CORDIC_WDATA or CORDIC_RDATA register. If an extraneous access to the relevant register occurs before this, the acknowledge is asserted prematurely, and may block the DMA channel. Therefore, when the DMA read channel is enabled, CPU access to the CORDIC_RDATA register must be avoided. Similarly, the processor must avoid accessing the CORDIC_WDATA register when the DMA write channel is enabled.

22.4 CORDIC registers

The CORDIC registers can only be accessed in 32-bit word format.

22.4.1 CORDIC control/status register (CORDIC_CSR)

Address offset: 0x00

Reset value: 0x0000 0050

<table>
<thead>
<tr>
<th>RRDY</th>
<th>RRDY</th>
<th>RRDY</th>
<th>RRDY</th>
<th>RRDY</th>
<th>RRDY</th>
<th>SIZE</th>
<th>SIZE</th>
<th>RES</th>
<th>DMA</th>
<th>DMA</th>
<th>DMA</th>
<th>IEN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **RRDY**: Result ready flag
- 0: No new data in output register
- 1: CORDIC_RDATA register contains new data.

This bit is set by hardware when a CORDIC operation completes. It is reset by hardware when the CORDIC_RDATA register is read (NRES+1) times.

When this bit is set, if the IEN bit is also set, the CORDIC interrupt is asserted. If the DMAREN bit is set, a DMA read channel request is generated. While this bit is set, no new calculation is started.

Bits 30:23 Reserved, must be kept at reset value.

Bit 22 **ARGSIZE**: Width of input data
- 0: 32-bit
- 1: 16-bit

ARGSIZE selects the number of bits used to represent input data.

If 32-bit data is selected, the CORDIC_WDATA register expects arguments in q1.31 format.

If 16-bit data is selected, the CORDIC_WDATA register expects arguments in q1.15 format.

The primary argument (ARG1) is written to the least significant half-word, and the secondary argument (ARG2) to the most significant half-word.

Bit 21 **RESSIZE**: Width of output data
- 0: 32-bit
- 1: 16-bit

RESSIZE selects the number of bits used to represent output data.

If 32-bit data is selected, the CORDIC_RDATA register contains results in q1.31 format.

If 16-bit data is selected, the least significant half-word of CORDIC_RDATA contains the primary result (RES1) in q1.15 format, and the most significant half-word contains the secondary result (RES2), also in q1.15 format.
Bit 20 **NARGS**: Number of arguments expected by the CORDIC\_WDATA register
0: Only one 32-bit write (or two 16-bit values if ARGSIZE = 1) is needed for the next calculation.
1: Two 32-bit values must be written to the CORDIC\_WDATA register to trigger the next calculation.
Reads return the current state of the bit.

Bit 19 **NRES**: Number of results in the CORDIC\_RDATA register
0: Only one 32-bit value (or two 16-bit values if RESSIZE = 1) is transferred to the CORDIC\_RDATA register on completion of the next calculation. One read from CORDIC\_RDATA resets the RRDY flag.
1: Two 32-bit values are transferred to the CORDIC\_RDATA register on completion of the next calculation. Two reads from CORDIC\_RDATA are necessary to reset the RRDY flag.
Reads return the current state of the bit.

Bit 18 **DMAWEN**: Enable DMA write channel
0: Disabled. No DMA write requests are generated.
1: Enabled. Requests are generated on the DMA write channel whenever no operation is pending.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 17 **DMAREN**: Enable DMA read channel
0: Disabled. No DMA read requests are generated.
1: Enabled. Requests are generated on the DMA read channel whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bit 16 **IEN**: Enable interrupt
0: Disabled. No interrupt requests are generated.
1: Enabled. An interrupt request is generated whenever the RRDY flag is set.
This bit is set and cleared by software. A read returns the current state of the bit.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SCALE[2:0]**: Scaling factor
The value of this field indicates the scaling factor applied to the arguments and/or results. A value n implies that the arguments have been multiplied by a factor $2^{-n}$, and/or the results need to be multiplied by $2^n$. Refer to Section 22.3.2 for the applicability of the scaling factor for each function and the appropriate range.

Bits 7:4 **PRECISION[3:0]**: Precision required (number of iterations)
0: reserved
1 to 15: (Number of iterations)/4
To determine the number of iterations needed for a given accuracy refer to Table 162.
Note that for most functions, the recommended range for this field is 3 to 6.
Bits 3:0 **FUNC[3:0]**: Function

- 0: Cosine
- 1: Sine
- 2: Phase
- 3: Modulus
- 4: Arctangent
- 5: Hyperbolic cosine
- 6: Hyperbolic sine
- 7: Arctanh
- 8: Natural logarithm
- 9: Square root
- 10 to 15: Reserved

### 22.4.2 CORDIC argument register (CORDIC_WDATA)

Address offset: 0x04

Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ARG[31:16]]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ARG[15:0]]</td>
</tr>
</tbody>
</table>

Bits 31:0 **ARG[31:0]**: Function input arguments

This register is programmed with the input arguments for the function selected in the CORDIC_CSR register FUNC field.

If 32-bit format is selected (CORDIC_CSR.ARGSIZE = 0) and two input arguments are required (CORDIC_CSR.NARGS = 1), two successive writes are required to this register. The first writes the primary argument (ARG1), the second writes the secondary argument (ARG2).

If 32-bit format is selected and only one input argument is required (NARGS = 0), only one write is required to this register, containing the primary argument (ARG1).

If 16-bit format is selected (CORDIC_CSR.ARGSIZE = 1), one write to this register contains both arguments. The primary argument (ARG1) is in the lower half, ARG[15:0], and the secondary argument (ARG2) is in the upper half, ARG[31:16]. In this case, NARGS must be set to 0.

Refer to Section 22.3.2 for the arguments required by each function, and their permitted range.

When the required number of arguments has been written, the CORDIC evaluates the function designated by CORDIC_CSR.FUNC using the supplied input arguments, provided any previous calculation has completed. If a calculation is ongoing, the ARG1 and ARG2 values are held pending until the calculation is completed and the results read. During this time, a write to the register cancels the pending operation and overwrites the argument data.
22.4.3 CORDIC result register (CORDIC_RDATA)

Address offset: 0x08
Reset value: 0x0000 0000

Bits 31:0 RES[31:0]: Function result

If 32-bit format is selected (CORDIC_CSR.RESSIZE = 0) and two output values are expected (CORDIC_CSR.NRES = 1), this register must be read twice when the RRDY flag is set. The first read fetches the primary result (RES1). The second read fetches the secondary result (RES2) and resets RRDY.

If 32-bit format is selected and only one output value is expected (NRES = 0), only one read of this register is required to fetch the primary result (RES1) and reset the RRDY flag.

If 16-bit format is selected (CORDIC_CSR.RESSIZE = 1), this register contains the primary result (RES1) in the lower half, RES[15:0], and the secondary result (RES2) in the upper half, RES[31:16]. In this case, NRES must be set to 0, and only one read performed. A read from this register resets the RRDY flag in the CORDIC_CSR register.

22.4.4 CORDIC register map

Table 163. CORDIC register map and reset value

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CORDIC_CSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>CORDIC_WDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ARG[31:0]</td>
<td>xxxxxxxxx</td>
<td>xxxxxxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>xx xxxx xx xx</td>
<td>xx xx xx xx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>CORDIC_RDATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RES[31:0]</td>
<td>xxxxxxxxx</td>
<td>xxxxxxxxx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
23 Flexible memory controller (FMC)

The flexible memory controller (FMC) includes three memory controllers:

- The NOR/PSRAM memory controller
- The NAND memory controller
- The Synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller

23.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:

- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/OneNAND flash memory
  - PSRAM (4 memory banks)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) memories
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16- or 32-bit wide data bus
- Independent Chip Select control for each memory bank
- Independent configuration for each memory bank
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control
- Write FIFO with 16 x32-bit depth
  The Write FIFO is common to all memory controllers and consists of:
  - a Write Data FIFO which stores the data to be written to the memory
  - a Write Address FIFO which stores the address (up to 28 bits) plus the data size (up to 2 bits). When operating in Burst mode, only the start address is stored except when crossing a page boundary (for PSRAM and SDRAM). In this case, the burst is broken into two FIFO entries.
- Cacheable Read FIFO with 6 x64-bit depth (6 x14-bit address tag) for SDRAM controller.

At startup the FMC pins must be configured by the user application. The FMC I/O pins which are not used by the application can be used for other purposes.
The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. However, only a few bits can be changed on-the-fly:

- ECCEN and PBKEN bits in the FMC_PCR register
- IFS, IRS and ILS bits in the FMC_SR register
- MODE[2:0], CTB1/CTB2, NRFS and MRD bits in the FMC_SDCMR register
- REIE and CRE bits in the FMC_SDRTR register.

Follow the below sequence to modify parameters while the FMC is enabled:

1. First disable the FMC to prevent further accesses to any memory controller while the register is modified.
2. Update all required configurations.
3. Enable the FMC again.

When the SDRAM controller is used, if the SDCLK Clock ratio or refresh rate has to be modified after initialization phase, the following procedure must be followed.

1. Put the SDRAM device in Self-refresh mode.
2. Disable the FMC by resetting the FMCEN bit in the FMC_BCR1 register.
3. Update the required parameters.
4. Enable the FMC once all parameters have been updated.
5. Then, send the Clock Configuration Enable command to exit Self-fresh mode.

### 23.2 FMC block diagram

The FMC consists of the following main blocks:

- The NOR flash/PSRAM/SRAM controller
- The NAND controller
- The SDRAM controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the figure below.
Figure 162. FMC block diagram

- fmc_it to NVIC
- fmc_hclk
- fmc_ker_ck
- 32-bit AHB bus
- 64-bit AXI bus

**NOR/PSRAM memory controller**
- FMC_NL (or NADV)
- FMC_CLK
- FMC_NBL[3:0]
- FMC_A[25:0]
- FMC_D[31:0]
- FMC_NOE
- FMC_NWE

**Shared signals**
- FMC_NCE
- FMC_INT

**NOR / SRAM shared signals**
- FMC_NWAIT

**NAND memory controller**
- FMC_NCE
- FMC_INT

**NOR / PSRAM / SRAM shared signals**
- FMC_NWE

**SDRAM controller**
- FMC_SDCLK
- FMC_SDNWE
- FMC_SDCKE[1:0]
- FMC_SDNE[1:0]
- FMC_NRAS
- FMC_NCAS

**NAND signals**
- FMC_NWAIT

**32-bit AHB bus**

**64-bit AXI bus**

**AXI/AHB interface & Configuration registers**
- fmc_hclk
- fmc_ker_ck

**NAND flash memory controller**
- FMC_NCE
- FMC_INT

**NOR / PSRAM shared signals**
- FMC_NWAIT

**NAND signals**
- FMC_NWAIT
23.3 **FMC internal signals**

*Table* 164 gives the list of FMC internal signals. FMC pins (or external signals) are described in *Section 23.7.1: External memory interface signals*.

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fmc_it</td>
<td>Digital output</td>
<td>FMC interrupt</td>
</tr>
<tr>
<td>fmc_ker_ck</td>
<td>Digital input</td>
<td>FMC kernel clock</td>
</tr>
<tr>
<td>fmc_hclk</td>
<td>Digital input</td>
<td>FMC interface clock</td>
</tr>
</tbody>
</table>

23.4 **AHB interface**

The AHB slave interface allows internal CPUs to configure the FMC registers. The AHB clock (fmc_hclk) is the reference clock for the FMC register accesses.

23.5 **AXI interface**

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32-, 16- or 8-bit accesses according to data size accesses. The FMC Chip Select (FMC_NEx) does not toggle between consecutive accesses except in case of accesses in mode D when the Extended mode is enabled.

The FMC generates an AXI slave error when one of the following conditions is met:

- Reading or writing to an FMC bank (Bank 1 to 4) which is not enabled.
- Writing to a write protected SDRAM bank (WP bit set in the FMC_SDCRx register).
- Violation of the SDRAM address range (access to reserved address range)
- Attempting to read/write access from/to SDRAM bank when it is not yet initialized

The FMC generates an AXI decoder error when ADDR[31:28] address bits are not supported by the FMC bank base address following the BMAP[1:0] bits configuration.

The kernel clock for the FMC is the asynchronous fmc_ker_ck clock (refer to *Section Reset and Clock control (RCC)* for fmc_ker_ck clock source selection).
### 23.5.1 Supported memories and transactions

#### General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. The best performance is always achieved with aligned AXI transactions whose size matches the external device data width.

When AXI transaction data size is different from the device data width, the result depends on the following factors:

- **AXI transaction data size is greater than the device data width:**
  - Read/Write transactions: the FMC splits the AXI transaction into smaller consecutive accesses matching the external device data width.

- **AXI transaction data size is smaller than the external device data width and the device supports byte selection (SRAM, PSRAM, SDRAM):**
  - Write transactions, the FMC manages the transaction using the byte lane signals.
  - Read transactions, the FMC returns all bytes according to the external device data width. The useless bytes are discarded by the system.

- **AXI transaction data size is smaller than the external device data width and the device does not support byte selection (NOR and NAND flash memories):**
  - Write transactions: the FMC writes some irrelevant bytes which may corrupt the external device.
  - Read transactions: the FMC returns all bytes according to the external device data width. The useless bytes are discarded by the system.

#### Caution: Address alignment

- Read transactions with unaligned addresses (such as half-word starting at an odd address) are not supported by the FMC.
- Write transactions with unaligned addresses:
  - Their support depends on byte selection availability on the external device:
    - If the device does not support byte selection (NOR and NAND flash memories), narrow write transactions and/or unaligned write transaction are not supported since the FMC would write irrelevant bytes and corrupt the external device.

#### Wrap support for NOR flash/PSRAM and SDRAM

The synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:

- The read is split into two linear burst transactions.
- The write is split into two linear burst transactions if the write FIFO is enabled and into several linear burst transactions if the write FIFO is disabled.

#### Configuration registers

The FMC can be configured through a set of registers. Refer to Section 23.7.6, for a detailed description of the NOR flash/PSRAM controller registers. Refer to Section 23.8.7, for a detailed description of the NAND flash registers and to Section 23.9.5 for a detailed description of the SDRAM controller registers.
23.6 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size banks of 256 Mbytes each (see Figure 163):

- Bank 1 is used to address up to 4 NOR flash memory or PSRAM devices. This bank is split into 4 NOR/PSRAM subbanks with 4 dedicated Chip Selects, as follows:
  - Bank 1 - NOR/PSRAM 1
  - Bank 1 - NOR/PSRAM 2
  - Bank 1 - NOR/PSRAM 3
  - Bank 1 - NOR/PSRAM 4
- Bank 2 and Bank 4 are not used by the FMC.
- Bank 3 is used to address NAND flash memory devices. The MPU memory attribute for this space must be reconfigured by software to Device.
- Bank 5 and 6 are used to address SDRAM devices (1 device per bank).

For each bank the type of memory to be used can be configured by the user application through the Configuration register.

Figure 163. FMC memory banks (default mapping)
The FMC bank mapping can be modified through the BMAP[1:0] bits in the FMC_BCR1 register. *Table 165* shows the configuration to swap the NOR/PSRAM bank with SDRAM banks.

### Table 165. FMC bank mapping options

<table>
<thead>
<tr>
<th>Start -End address</th>
<th>BMAP[1:0]=00 (Default mapping)</th>
<th>BMAP[1:0]=01 NOR/PSRAM and SDRAM banks swapped</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x6000 0000 - 0x6FFF FFFF</td>
<td>NOR/PSRAM bank</td>
<td>SDRAM bank1</td>
</tr>
<tr>
<td>0x7000 0000 - 0x7FFF FFFF</td>
<td>Not used by FMC</td>
<td></td>
</tr>
<tr>
<td>0x8000 0000 - 0x8FFF FFFF</td>
<td>NAND bank</td>
<td>NAND bank</td>
</tr>
<tr>
<td>0x9000 0000 - 0x9FFF FFFF</td>
<td>Not used by FMC</td>
<td></td>
</tr>
<tr>
<td>0xC000 0000 - 0xCFFF FFFF</td>
<td>SDRAM bank1</td>
<td>NOR/PSRAM bank</td>
</tr>
<tr>
<td>0xD000 0000 - 0xDFFF FFFF</td>
<td>SDRAM bank2</td>
<td>SDRAM bank2</td>
</tr>
</tbody>
</table>

#### 23.6.1 NOR/PSRAM address mapping

ADDR[27:26] bits are used to select one of the four memory banks as shown in *Table 166*.

### Table 166. NOR/PSRAM bank selection

<table>
<thead>
<tr>
<th>ADDR<a href="1">27:26</a></th>
<th>Selected bank</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Bank 1 - NOR/PSRAM 1</td>
</tr>
<tr>
<td>01</td>
<td>Bank 1 - NOR/PSRAM 2</td>
</tr>
<tr>
<td>10</td>
<td>Bank 1 - NOR/PSRAM 3</td>
</tr>
<tr>
<td>11</td>
<td>Bank 1 - NOR/PSRAM 4</td>
</tr>
</tbody>
</table>

1. ADDR are internal address lines that are translated to external memory.

The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

### Table 167. NOR/PSRAM External memory address

<table>
<thead>
<tr>
<th>Memory width(1)</th>
<th>Data address issued to the memory</th>
<th>Maximum memory capacity (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit</td>
<td>ADDR[25:0]</td>
<td>64 Mbytes x 8 = 512 Mbit</td>
</tr>
<tr>
<td>16-bit</td>
<td>ADDR[25:1] &gt;&gt; 1</td>
<td>64 Mbytes/2 x 16 = 512 Mbit</td>
</tr>
<tr>
<td>32-bit</td>
<td>ADDR[25:2] &gt;&gt; 2</td>
<td>64 Mbytes/4 x 32 = 512 Mbit</td>
</tr>
</tbody>
</table>

1. In case of a 16-bit external memory width, the FMC internally uses ADDR[25:1] to generate the address for external memory FMC_A[24:0]. In case of a 32-bit memory width, the FMC internally uses ADDR[25:2] to generate the external address. Whatever the external memory width, FMC_A[0] should be connected to external memory address A[0].

#### 23.6.2 NAND flash memory address mapping

The NAND bank is divided into memory areas as indicated in *Table 168*. 
For NAND flash memory, the common and attribute memory spaces are subdivided into three sections (see in Table 169 below) located in the lower 256 Kbytes:

- Data section (first 64 Kbytes in the common/attribute memory space)
- Command section (second 64 Kbytes in the common/attribute memory space)
- Address section (next 128 Kbytes in the common/attribute memory space)

The application software uses the 3 sections to access the NAND flash memory:
- To send a command to NAND flash memory, the software must write the command value to any memory location in the command section.
- To specify the NAND flash address that must be read or written, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- To read or write data, the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

### 23.6.3 SDRAM address mapping

Two SDRAM banks are available as indicated in Table 170.

<table>
<thead>
<tr>
<th>Table 170. SDRAM bank selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected bank</td>
</tr>
<tr>
<td>SDRAM Bank1</td>
</tr>
<tr>
<td>SDRAM Bank2</td>
</tr>
</tbody>
</table>

Table 171 shows SDRAM mapping for a 13-bit row and an 11-bit column configuration.
The ADDR[27:0] bits are translated into an external SDRAM address depending on the SDRAM controller configuration:

- Data size: 8, 16 or 32 bits
- Row size: 11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks

The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

### Table 171. SDRAM address mapping

<table>
<thead>
<tr>
<th>Memory width(1)</th>
<th>Internal bank</th>
<th>Row address</th>
<th>Column address (2)</th>
<th>Maximum memory capacity (Mbytes)</th>
</tr>
</thead>
</table>

1. When interfacing with a 16-bit memory, the FMC internally uses the ADDR[11:1] internal address lines to generate the external address. When interfacing with a 32-bit memory, the FMC internally uses ADDR[12:2] lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the external memory address A[0].

2. The AutoPrecharge is not supported. FMC_A[10] must be connected to the external memory address A[10] but it is always driven low.

### Table 172. SDRAM address mapping with 8-bit data bus width(1)(2)

<table>
<thead>
<tr>
<th>Row size configuration</th>
<th>ADDR (Internal Address Lines)</th>
<th>27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>configuration</td>
<td>Res. Bank [1:0]</td>
<td>Row[10:0]</td>
</tr>
</tbody>
</table>
### Table 172. SDRAM address mapping with 8-bit data bus width$^{(1,2)}$ (continued)

<table>
<thead>
<tr>
<th>Row size configuration</th>
<th>ADDR(Internal Address Lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved (Res.) address range generates an AXI slave error.

### Table 173. SDRAM address mapping with 16-bit data bus width$^{(1,2)}$

<table>
<thead>
<tr>
<th>Row size Configuration</th>
<th>ADDR(address Lines)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>11-bit row size configuration</td>
<td>Res. Bank [1:0] Row[10:0] Column[7:0] BM0 (3)</td>
</tr>
<tr>
<td></td>
<td>Res. Bank [1:0] Row[10:0] Column[8:0] BM0</td>
</tr>
</tbody>
</table>

922/3791  RM0477 Rev 6
Table 173. SDRAM address mapping with 16-bit data bus width\(^{(1)(2)}\)

| Row size          | ADDR(address Lines) | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|-------------------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| configuration     | Res.                | Bank [1:0] | Row[12:0] | Column[8:0] | BM0 |
| configuration      | Res.                | Bank [1:0] | Row[10:0] | Column[8:0] | BM0 |
| configuration      | Res.                | Bank [1:0] | Row[12:0] | Column[8:0] | BM0 |

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AXI Slave error.
3. BM0: is the byte mask for 16-bit access.

Table 174. SDRAM address mapping with 32-bit data bus width\(^{(1)(2)}\)

| Row size          | ADDR(address Lines) | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|-------------------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be set to ‘0’.
2. Access to Reserved space (Res.) generates an AXI Slave error.
3. BM[1:0]: is the byte mask for 32-bit access.
23.7 NOR flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM and ROM
  - 8 bits
  - 16 bits
  - 32 bits
- PSRAM (Cellular RAM)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per bank. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a sub-multiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, Bank 1 must be configured in Synchronous mode (see Section 23.7.6: NOR/PSRAM controller registers). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID) otherwise the FMC_CLK frequency is changed depending on AXI data transaction (refer to Section 23.7.5: Synchronous transactions for FMC_CLK divider ratio formula).

The size of each bank is fixed and equal to 64 Mbytes. Each bank is configured through dedicated registers (see Section 23.7.6: NOR/PSRAM controller registers).

The programmable memory parameters include access times (see Table 175) and support for wait management (for PSRAM and NOR flash memory accessed in Burst mode).
23.7.1 External memory interface signals

Table 176, Table 177 and Table 178 list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix “N” identifies the signals which are active low.

NOR flash memory, non-multiplexed I/Os

Table 176. Non-multiplexed I/O NOR flash memory

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>A[25:0]</td>
<td>O</td>
<td>Address bus</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>I/O</td>
<td>Bidirectional data bus</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Latch enable (this signal is called address valid, NADV, by some NOR flash devices)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>NOR flash wait input signal to the FMC</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits (26 address lines).
Flexible memory controller (FMC) RM0477

NOR flash memory, 16-bit multiplexed I/Os

Table 177. 16-bit multiplexed I/O NOR flash memory

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>AD[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Latch enable (this signal is called address valid, NADV, by some NOR flash devices)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>NOR flash wait input signal to the FMC</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits.

PSRAM/SRAM, non-multiplexed I/Os

Table 178. Non-multiplexed I/Os PSRAM/SRAM

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (only for PSRAM synchronous access)</td>
</tr>
<tr>
<td>A[25:0]</td>
<td>O</td>
<td>Address bus</td>
</tr>
<tr>
<td>D[31:0]</td>
<td>I/O</td>
<td>Data bidirectional bus</td>
</tr>
<tr>
<td>NE[x]</td>
<td>O</td>
<td>Chip Select, x = 1..4 (called NCE by PSRAM (Cellular RAM i.e. CRAM))</td>
</tr>
<tr>
<td>NOE</td>
<td>O</td>
<td>Output enable</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NL(=NADV)</td>
<td>O</td>
<td>Address valid only for PSRAM input (memory signal name: NADV)</td>
</tr>
<tr>
<td>NWAIT</td>
<td>I</td>
<td>PSRAM wait input signal to the FMC</td>
</tr>
<tr>
<td>NBL[3:0]</td>
<td>O</td>
<td>Byte lane output. Byte 0 to Byte 3 control (Upper and lower byte enable)</td>
</tr>
</tbody>
</table>

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 179. 16-Bit multiplexed I/O PSRAM

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>O</td>
<td>Clock (for synchronous access)</td>
</tr>
<tr>
<td>AD[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)</td>
</tr>
</tbody>
</table>
The maximum capacity is 512 Mbits (26 address lines).

23.7.2 Supported memories and transactions

Table 180 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/ not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOR flash (muxed I/Os and non-multiplexed I/Os)</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous page</td>
<td>R</td>
<td>-</td>
<td>16</td>
<td>N</td>
<td>Mode is not supported</td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Synchronous</td>
<td>R</td>
<td>32/64</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
</tbody>
</table>
23.7.3 General timing rules

Signal synchronization is performed as follows:

- All controller output signals change on the rising edge of the fmc_ker_ck clock.
- In Synchronous read and write modes, all output signals change on the rising edge of fmc_ker_ck clock. Whatever the CLKDIV value, all outputs change as follows:
  - NOEL/NWEL/ NEL/NADVL/ NADVH /NBLL/ Address valid outputs change on the falling edge of FMC_CLK clock.
  - NOEH/ NWEH / NEH/ NOEH/NBLH/ Address invalid outputs change on the rising edge of FMC_CLK clock.

---

### Table 180. NOR flash/PSRAM: Example of supported memories and transactions\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSRAM (multiplexed I/Os and non-multiplexed I/Os)</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>64</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>-</td>
<td>16</td>
<td>N</td>
<td>Mode is not supported</td>
</tr>
<tr>
<td>Synchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous</td>
<td>R</td>
<td>32/64</td>
<td>16</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td></td>
<td>Use of byte lanes NBL[1:0]</td>
</tr>
<tr>
<td>Synchronous</td>
<td>W</td>
<td>16/32/64</td>
<td>16</td>
<td>Y</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SRAM and ROM

| Asynchronous | R | 8/16 | 16 | Y | Use of byte lanes NBL[1:0] |
| Asynchronous | W | 8/16 | 16 | Y | Use of byte lanes NBL[1:0] |
| Asynchronous | R | 32  | 16 | Y | Split into 2 FMC accesses |
| Asynchronous | W | 32  | 16 | Y | Split into 2 FMC accesses |
| Asynchronous | R | 64  | 16 | Y | Split into 4 FMC accesses |
| Asynchronous | W | 64  | 16 | Y | Split into 4 FMC accesses |

---

1. NBL[1:0] are also driven by AXI write strobes.
23.7.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous transactions on static memories (NOR flash memory, PSRAM, SRAM) are performed as follows:

- Signals are synchronized by the internal clock. This clock is not issued to the memory.
- The FMC always samples the data before deasserting the Chip Select signal. This guarantees that the memory data hold timing constraint is met (minimum Chip Enable high to data transition is usually 0 ns)
- If the Extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B.
- If the Extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in Mode1 or Mode2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 164. Mode 1 read access waveforms
The fmc_ker_ck cycle at the end of the write transaction helps guarantee the address and data hold time after the NWE rising edge. Due to the presence of this fmc_ker_ck cycle, the DATAST value must be greater than zero (DATAST > 0).

Table 181. FMC_BCRx bitfields (mode 1)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
</tbody>
</table>
### Table 181. FMC_BCRx bitfields (mode 1) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Don’t care</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed, exclude 0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXE</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 182. FMC_BTRx bitfields (mode 1)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN (fmc_ker_ck))</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase ((DATAST+1) (fmc_ker_ck) cycles for write accesses, (DATAST) (fmc_ker_ck) cycles for read accesses).</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase ((ADDSET) (fmc_ker_ck) cycles). Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode A - SRAM/PSRAM (CRAM) OE toggling

Figure 166. Mode A read access waveforms

1. NBL[3:0] are driven low during the read access
The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

Table 183. FMC_BCRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
</tbody>
</table>
Table 183. FMC_BCRx bitfields (mode A) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCE</td>
<td>Don’t care</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed, exclude 0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

Table 184. FMC_BTRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

Table 185. FMC_BWTRx bitfields (mode A)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode 2/B - NOR flash

Figure 168. Mode 2 and mode B read access waveforms

Figure 169. Mode 2 write access waveforms
The differences with Mode1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 186. FMC_BCRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1 for mode B, 0x0 for mode 2</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
</tbody>
</table>
### Table 186. FMC_BCRx bitfields (mode 2/B) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 187. FMC_BTRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x1 if Extended mode is set</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

### Table 188. FMC_BWTRx bitfields (mode 2/B)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x1 if Extended mode is set</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Note: The FMC_BWTRx register is valid only if the Extended mode is set (mode B), otherwise its content is don’t care.

Mode C - NOR flash - OE toggling

**Figure 171. Mode C read access waveforms**

**Figure 172. Mode C write access waveforms**
The differences compared with Mode1 are the toggling of NOE and the independent read and write timings.

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x02 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>
### Table 190. FMC_BTRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x2</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>0x0</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>0x0</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>

### Table 191. FMC_BWTRx bitfields (mode C)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x2</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.</td>
</tr>
</tbody>
</table>
Mode D - asynchronous access with extended address

Figure 173. Mode D read access waveforms

Figure 174. Mode D write access waveforms
The differences with Mode1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 192. FMC_BCRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x1</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don’t care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>As needed</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x0</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

Table 193. FMC_BTRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x3</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.</td>
</tr>
</tbody>
</table>
### Table 193. FMC_BTRx bitfields (mode D) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>

### Table 194. FMC_BWTRx bitfields (mode D)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x3</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don’t care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don’t care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.</td>
</tr>
</tbody>
</table>
Muxed mode - multiplexed asynchronous access to NOR flash memory

The difference with Mode D is the drive of the lower address byte(s) on the data bus.
### Table 195. FMC_BCRx bitfields (Muxed mode)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>Set to 1 if the memory supports this feature. Otherwise keep at 0.</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>0x0 (no effect in Asynchronous mode)</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>As needed</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>Don't care</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>Meaningful only if bit 15 is 1</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x0</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x2 (NOR flash memory)</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>0x1</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 196. FMC_BTRx bitfields (Muxed mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Don't care</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>Don't care</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).</td>
</tr>
</tbody>
</table>
WAIT management in asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 fmc_ker_ck cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:
   
   \[ \text{DATAST} \geq (4 \times \text{fmc_ker_ck}) + \text{max_wait_assertion_time} \]

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   
   if
   
   \[ \text{max_wait_assertion_time} > \text{address_phase} + \text{hold_phase} \]
   
   then:
   
   \[ \text{DATAST} \geq (4 \times \text{fmc_ker_ck}) + (\text{max_wait_assertion_time} - \text{address_phase} - \text{hold_phase}) \]

   otherwise
   
   \[ \text{DATAST} \geq (4 \times \text{fmc_ker_ck}) \]

   where max_wait Assertion time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

*Figure 177* and *Figure 178* show the number of fmc_ker_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).
Figure 177. Asynchronous wait during a read access waveforms

1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 178. Asynchronous wait during a write access waveforms

1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.
23.7.5 **Synchronous transactions**

The memory clock, FMC_CLK, is a sub-multiple of fmc_ker_ck. It depends on the value of CLKDIV and the MWID/AXI data size, following the formula given below:

\[
\text{FMC_CLK divider ratio} = \max(\text{CLKDIV} + 1, \text{MWID}(\text{AXI data size}))
\]

If MWID is 16 or 8-bit, the FMC_CLK divider ratio is always defined by the programmed CLKDIV value.

If MWID is 32-bit, the FMC_CLK divider ratio depends also on AXI data size.

Example:
- If CLKDIV=1, MWID = 32 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/4.
- If CLKDIV=1, MWID = 16 bits, AXI data size=8 bits, FMC_CLK=fmc_ker_ck/2.

NOR flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

For some PSRAM memories which must be configured to Synchronous mode, during the BCR register writing, the memory attribute space must be configured to device or strongly-ordered. Once PSRAM BCR register is configured, the memory attribute of PSRAM address space can be programmed to cacheable.

**Data latency versus NOR memory latency**

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

**Caution:** Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:
- NOR flash latency = (DATLAT + 2) FMC_CLK clock cycles
- or NOR flash latency = (DATLAT + 3) FMC_CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.
**Single-burst transfer**

When the selected bank is configured in Burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and deassert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access would first require to re-program the memory access mode, which would altogether last longer.

**Cross boundary page for Cellular RAM 1.5**

Cellular RAM 1.5 does not allow burst access to cross the page boundary. The FMC allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

**Wait management**

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In Burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait state (default after reset).
- The flash memory asserts the NWAIT signal during the wait state

The FMC supports both NOR flash wait state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0..3).
1. Byte lane outputs (NBL) are not shown; for NOR access, they are held high, and, for PSRAM (CRAM) access, they are held low.
### Table 197. FMC_BCRx bitfields (Synchronous multiplexed read mode)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>MC</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>No effect on synchronous read</td>
</tr>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>As needed. (0x1 when using CRAM 1.5)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCWAIT</td>
<td>0x0</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>to be set to 1 if the memory supports this feature, to be kept at 0 otherwise</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>no effect on synchronous read</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>0x1</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support (NOR flash memory)</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x1 or 0x2</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>As needed</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

### Table 198. FMC_BTRx bitfields (Synchronous multiplexed read mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
<tr>
<td>23:20</td>
<td>CLKDIV</td>
<td>0x0 to get CLK = fmc_ker_ck</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x1 to get CLK = 2 × fmc_ker_ck</td>
</tr>
<tr>
<td></td>
<td></td>
<td>..</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>
Table 198. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>

Figure 181. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)

1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.
2. Byte Lane (NBL) outputs are not shown, they are held low while NEx is active.

Table 199. FMC_BCRx bitfields (Synchronous multiplexed write mode)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>FMCEN</td>
<td>0x1</td>
</tr>
<tr>
<td>30:26</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>25:24</td>
<td>BMAP</td>
<td>As needed</td>
</tr>
<tr>
<td>23:22</td>
<td>Reserved</td>
<td>0x000</td>
</tr>
<tr>
<td>21</td>
<td>WFDIS</td>
<td>As needed</td>
</tr>
<tr>
<td>20</td>
<td>CCLKEN</td>
<td>As needed</td>
</tr>
<tr>
<td>19</td>
<td>CBURSTRW</td>
<td>No effect on synchronous read</td>
</tr>
</tbody>
</table>
Table 199. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

<table>
<thead>
<tr>
<th>Bit No.</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>18:16</td>
<td>CPSIZE</td>
<td>As needed. (0x1 when using CRAM 1.5)</td>
</tr>
<tr>
<td>15</td>
<td>ASYNCEWAIT</td>
<td>0x0</td>
</tr>
<tr>
<td>14</td>
<td>EXTMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>13</td>
<td>WAITEN</td>
<td>to be set to 1 if the memory supports this feature, to be kept at 0 otherwise.</td>
</tr>
<tr>
<td>12</td>
<td>WREN</td>
<td>0x1</td>
</tr>
<tr>
<td>11</td>
<td>WAITCFG</td>
<td>0x0</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>9</td>
<td>WAITPOL</td>
<td>to be set according to memory</td>
</tr>
<tr>
<td>8</td>
<td>BURSTEN</td>
<td>no effect on synchronous write</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>0x1</td>
</tr>
<tr>
<td>6</td>
<td>FACCEN</td>
<td>Set according to memory support</td>
</tr>
<tr>
<td>5:4</td>
<td>MWID</td>
<td>As needed</td>
</tr>
<tr>
<td>3:2</td>
<td>MTYP</td>
<td>0x1</td>
</tr>
<tr>
<td>1</td>
<td>MUXEN</td>
<td>As needed</td>
</tr>
<tr>
<td>0</td>
<td>MBKEN</td>
<td>0x1</td>
</tr>
</tbody>
</table>

Table 200. FMC_BTRx bitfields (Synchronous multiplexed write mode)

<table>
<thead>
<tr>
<th>Bit number</th>
<th>Bit name</th>
<th>Value to set</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:30</td>
<td>Reserved</td>
<td>0x0</td>
</tr>
<tr>
<td>29:28</td>
<td>ACCMOD</td>
<td>0x0</td>
</tr>
<tr>
<td>27:24</td>
<td>DATLAT</td>
<td>Data latency</td>
</tr>
<tr>
<td>23:20</td>
<td>CLDIV</td>
<td>0x0 to get CLK = fmc_ker_ck 0x1 to get CLK = 2 × fmc_ker_ck</td>
</tr>
<tr>
<td>19:16</td>
<td>BUSTURN</td>
<td>Time between NEx high to NEx low (BUSTURN fmc_ker_ck)</td>
</tr>
<tr>
<td>15:8</td>
<td>DATAST</td>
<td>Don’t care</td>
</tr>
<tr>
<td>7:4</td>
<td>ADDHLD</td>
<td>Don’t care</td>
</tr>
<tr>
<td>3:0</td>
<td>ADDSET</td>
<td>Don’t care</td>
</tr>
</tbody>
</table>
23.7.6  NOR/PSRAM controller registers

SRAM/NOR-flash chip-select control registers for bank x (FMC_BCRx)

Address offset: 0x00 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, Block 3: 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR flash memories.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>FMCEN</th>
<th>Bits 30:26</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FMCEN</td>
<td>FMC Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: Disable the FMC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Enable the FMC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: The FMCEN bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 25:24</th>
<th>BMAP[1:0]: FMC bank mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits allow different remap or swap of the FMC NOR/PSRAM and SDRAM banks (refer to Table 165).</td>
<td></td>
</tr>
<tr>
<td>00: Default mapping (refer to Figure 163 and Table 165).</td>
<td></td>
</tr>
<tr>
<td>01: NOR/PSRAM bank and SDRAM bank 1 are swapped.</td>
<td></td>
</tr>
<tr>
<td>10: Reserved</td>
<td></td>
</tr>
<tr>
<td>11: Reserved.</td>
<td></td>
</tr>
<tr>
<td>Note: The BMAP bits of the FMC_BCR2..4 registers are don’t care. It is only enabled through the FMC_BCR1 register.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 21</th>
<th>WFDIS: Write FIFO Disable</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit disables the Write FIFO used by the FMC.</td>
<td></td>
</tr>
<tr>
<td>0: Write FIFO enabled (Default after reset)</td>
<td></td>
</tr>
<tr>
<td>1: Write FIFO disabled</td>
<td></td>
</tr>
<tr>
<td>Note: The WFDIS bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 20 CCLKEN: Continuous Clock Enable
This bit enables the FMC_CLK clock output to external memory devices.
0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is specified by the programmed CLKDIV value in the FMC_BCRx register (default after reset).
1: The FMC_CLK is generated continuously during asynchronous and synchronous access. The FMC_CLK clock is activated when the CCLKEN is set.

Note: The CCLKEN bit of the FMC_BCR2..4 registers is don’t care. It is only enabled through the FMC_BCR1 register. Bank 1 must be configured in Synchronous mode to generate the FMC_CLK continuous clock.

If CCLKEN bit is set, the FMC_CLK clock ratio is specified by CLKDIV value in the FMC_BTR1 register. CLKDIV in FMC_BWTR1 is don’t care.

If the Synchronous mode is used and CCLKEN bit is set, the synchronous memories connected to other banks than Bank 1 are clocked by the same clock (the CLKDIV value in the FMC_BTR2..4 and FMC_BWTR2..4 registers for other banks has no effect.)

Bit 19 CBURSTR: Write burst enable
For PSRAM (CRAM) operating in Burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.
0: Write operations are always performed in Asynchronous mode
1: Write operations are performed in Synchronous mode.

Bits 18:16 CPSIZE[2:0]: CRAM Page Size
These are used for Cellular RAM 1.5 which does not allow burst access to cross the address boundaries between pages. When these bits are configured, the FMC splits automatically the burst access when the memory page size is reached (refer to memory datasheet for page size).
000: No burst split when crossing page boundary (default after reset).
001: 128 bytes
010: 256 bytes
100: 1024 bytes
Other configuration: reserved.

Bit 15 ASYNCHWAIT: Wait signal during asynchronous transfers
This bit enables/disables the FMC to use the wait signal even during an asynchronous protocol.
0: NWAIT signal is not taken in to account when running an asynchronous protocol (default after reset)
1: NWAIT signal is taken in to account when running an asynchronous protocol

Bit 14 EXTMOD: Extended mode enable.
This bit enables the FMC to program the write timings for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.
0: values inside FMC_BWTR register are not taken into account (default after reset)
1: values inside FMC_BWTR register are taken into account

Note: When the Extended mode is disabled, the FMC can operate in Mode1 or Mode2 as follows:
- Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP = 0x0 or 0x01)
- Mode 2 is the default mode when the NOR memory type is selected (MTYP = 0x10).
Bit 13  **WAITEN**: Wait enable bit  
This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in Synchronous mode.  
0: NWAIT signal is disabled (its level not taken into account, no wait state inserted after the programmed flash latency period)  
1: NWAIT signal is enabled (its level is taken into account after the programmed latency period to insert wait states if asserted) (default after reset)  

Bit 12  **WREN**: Write enable bit  
This bit indicates whether write operations are enabled/disabled in the bank by the FMC:  
0: Write operations are disabled in the bank by the FMC, an AXI slave error is reported  
1: Write operations are enabled for the bank by the FMC (default after reset).  

Bit 11  **WAITCFG**: Wait timing configuration  
The NWAIT signal indicates whether the data from the memory are valid or if a wait state must be inserted when accessing the memory in Synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait state or during the wait state:  
0: NWAIT signal is active one data cycle before wait state (default after reset)  
1: NWAIT signal is active during wait state (not used for PSRAM).  

Bit 10  Reserved, must be kept at reset value.  

Bit 9  **WAITPOL**: Wait signal polarity bit  
This bit defines the polarity of the wait signal from memory used for either in Synchronous or Asynchronous mode:  
0: NWAIT active low (default after reset)  
1: NWAIT active high.  

Bit 8  **BURSTEN**: Burst enable bit  
This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in Burst mode:  
0: Burst mode disabled (default after reset). Read accesses are performed in Asynchronous mode  
1: Burst mode enable. Read accesses are performed in Synchronous mode.  

Bit 7  Reserved, must be kept at reset value.  

Bit 6  **FACCEN**: Flash access enable  
This bit enables NOR flash memory access operations.  
0: Corresponding NOR flash memory access is disabled  
1: Corresponding NOR flash memory access is enabled (default after reset)  

Bits 5:4  **MWID[1:0]**: Memory data bus width  
Defines the external memory device width, valid for all type of memories.  
00: 8 bits  
01: 16 bits (default after reset)  
10: 32 bits  
11: reserved
Bits 3:2 **MTYP[1:0]: Memory type**

These bits define the type of external memory attached to the corresponding memory bank:
- 00: SRAM (default after reset for Bank 2...4)
- 01: PSRAM (CRAM)
- 10: NOR flash/OneNAND flash (default after reset for Bank 1)
- 11: reserved

Bit 1 **MUXEN: Address/data multiplexing enable bit**

When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
- 0: Address/Data non-multiplexed
- 1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN: Memory bank enable bit**

This bit enables the memory bank. After reset Bank1 is enabled, all others are disabled. Accessing a disabled bank causes an ERROR on AXI bus.
- 0: Corresponding memory bank is disabled
- 1: Corresponding memory bank is enabled
SRAM/NOR-flash chip-select timing registers for bank x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank, used for SRAMs, PSRAM and NOR flash memories. If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, 2 registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw</td>
<td>rw rw</td>
<td>rw rw</td>
<td>rw rw</td>
<td>rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>DATAST[7:0]</td>
<td>ADDHLD[3:0]</td>
<td>ADDSET[3:0]</td>
<td></td>
</tr>
<tr>
<td>rw rw</td>
<td>rw rw</td>
<td>rw rw</td>
<td>rw rw</td>
<td>rw rw</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ACCMOD[1:0]:** Access mode

These bits specify the Asynchronous access modes as shown in the timing diagrams. They are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: access mode A
01: access mode B
10: access mode C
11: access mode D

Bits 27:24 **DATLAT[3:0]:** (see note below bit descriptions): Data latency for synchronous memory

For synchronous access with read/write Burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data:

This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.

For asynchronous access, this value is don't care.

0000: Data latency of 2 FMC_CLK clock cycles for first burst access
1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

These bits define the period of FMC_CLK clock output signal, expressed in number of fmc_ker_ck cycles:

0000: FMC_CLK period = 1 x fmc_ker_ck period
0001: FMC_CLK period = 2 x fmc_ker_ck periods
0010: FMC_CLK period = 3 x fmc_ker_ck periods
1111: FMC_CLK period = 16 x fmc_ker_ck periods (default value after reset)

In asynchronous NOR flash, SRAM or PSRAM accesses, this value is don't care.

*Note: Refer to Section 23.7.5: Synchronous transactions for FMC_CLK divider ratio formula*
Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write-to-read (and read-to-write) transaction. This delay allows to match the minimum time between consecutive transactions ($t_{EHEL}$ from NEx high to NEx low) and the maximum time needed by the memory to free the data bus after a read access ($t_{EHQZ}$). The programmed bus turnaround delay is inserted between an asynchronous read (muxed or mode D) or write transaction and any other asynchronous/synchronous read or write to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different except for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 1 FMC clock cycle between:
  - Two consecutive asynchronous read transfers to the same static memory bank except for muxed and D modes.
  - An asynchronous read to an asynchronous or synchronous write to any static bank or dynamic bank except for muxed and D modes.
  - An asynchronous (modes 1, 2, A, B or C) read and a read from another static bank.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - Two consecutive synchronous writes (burst or single) to the same bank.
  - A synchronous write (burst or single) access and an asynchronous write or read transfer to or from static memory bank (the bank can be the same or different for the case of read).
  - Two consecutive synchronous reads (burst or single) followed by any synchronous/asynchronous read or write from/to another static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - Two consecutive synchronous writes (burst or single) to different static bank.
  - A synchronous write (burst or single) access and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

... 
1111: BUSTURN phase duration = 15 x fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]**: Data-phase duration

These bits are written by software to define the duration of the data phase (refer to Figure 164 to Figure 176), used in asynchronous accesses:

0000 0000: Reserved
0000 0001: DATAST phase duration = 1 x fmc_ker_ck clock cycles
0000 0010: DATAST phase duration = 2 x fmc_ker_ck clock cycles

... 
1111 1111: DATAST phase duration = 255 x fmc_ker_ck clock cycles (default value after reset)

For each memory type and access mode data-phase duration, refer to the respective figure (Figure 164 to Figure 176).

Example: Mode1, write access, DATAST = 1: Data-phase duration = DATAST+1 = 1 x fmc_ker_ck clock cycles.

Note: In synchronous accesses, this value is don’t care.
Bits 7:4 ADDHLD[3:0]: Address-hold phase duration
These bits are written by software to define the duration of the address hold phase (refer to Figure 164 to Figure 176), used in mode D or multiplexed accesses:
- 0000: Reserved
- 0001: ADDHLD phase duration = 1 × fmc_ker_ck clock cycle
- 0010: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle
  ... 
- 1111: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)
For each access mode address-hold phase duration, refer to the respective figure (Figure 164 to Figure 176).

Note: In synchronous accesses, this value is not used, the address hold phase is always 1 memory clock period duration.

Bits 3:0 ADDSET[3:0]: Address setup phase duration
These bits are written by software to define the duration of the address setup phase (refer to Figure 164 to Figure 176), used in SRAMs, ROMs and asynchronous NOR flash:
- 0000: ADDSET phase duration = 0 × fmc_ker_ck clock cycle
  ... 
- 1111: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)
For each access mode address setup phase duration, refer to the respective figure (refer to Figure 164 to Figure 176).

Note: In synchronous accesses, this value is don't care.
In Muxed mode or mode D, the minimum value for ADDSET is 1.
In mode 1 and PSRAM memory, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed.

On PSRAMs (CRAMs) the filled DATLAT must be set to 0, so that the FMC exits its latency phase soon and starts sampling NWAIT from memory, then starts to read or write when the memory is ready.

This method can be used also with the latest generation of synchronous flash memories that issue the NWAIT signal, unlike older flash memories (check the datasheet of the specific flash memory being used).
SRAM/NOR-flash write timing registers for bank x (FMC_BWTRx)

Address offset: 0x104 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory bank. It is used for SRAMs, PSRAMs and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACCMOD[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ACCMOD[1:0]**: Access mode.

These bits specify the asynchronous access modes as shown in the next timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

- 00: access mode A
- 01: access mode B
- 10: access mode C
- 11: access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay at the end of a write transaction to match the minimum time between consecutive transactions (tEHEL from ENx high to ENx low):

(BUSTURN + 1) fmc_ker_ck period ≥ tEHELmin

The programmed bus turnaround delay is inserted between an asynchronous write transfer and any other asynchronous/synchronous read or write transfer to or from a static bank. The bank can be the same or different in case of read, in case of write the bank can be different except for muxed or mode D.

In some cases, whatever the programmed BUSTURN values, the bus turnaround delay is fixed as follows:

- The bus turnaround delay is not inserted between two consecutive asynchronous write transfers to the same static memory bank except for muxed and D modes.
- There is a bus turnaround delay of 2 FMC clock cycle between:
  - Two consecutive synchronous writes (burst or single) to the same bank.
  - A synchronous write (burst or single) transfer and an asynchronous write or read transfer to or from static memory bank.
- There is a bus turnaround delay of 3 FMC clock cycle between:
  - Two consecutive synchronous writes (burst or single) to different static bank.
  - A synchronous write (burst or single) transfer and a synchronous read from the same or a different bank.

0000: BUSTURN phase duration = 0 fmc_ker_ck clock cycle added

... 

1111: BUSTURN phase duration = 15 fmc_ker_ck clock cycles added (default value after reset)
Bits 15:8 DATAST[7:0]: Data-phase duration.
These bits are written by software to define the duration of the data phase (refer to Figure 164 to Figure 176), used in asynchronous SRAM, PSRAM and NOR flash memory accesses:
- 0000 0000: Reserved
- 0000 0001: DATAST phase duration = 1 × fmc_ker_ck clock cycles
- 0000 0010: DATAST phase duration = 2 × fmc_ker_ck clock cycles
- ... 1111 1111: DATAST phase duration = 255 × fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 ADDHLD[3:0]: Address-hold phase duration.
These bits are written by software to define the duration of the address hold phase (refer to Figure 164 to Figure 176), used in asynchronous multiplexed accesses:
- 0000: Reserved
- 0001: ADDHLD phase duration = 1 × fmc_ker_ck clock cycle
- 0010: ADDHLD phase duration = 2 × fmc_ker_ck clock cycle
- ... 1111: ADDHLD phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous NOR flash accesses, this value is not used, the address hold phase is always 1 flash clock period duration.

Bits 3:0 ADDSET[3:0]: Address setup phase duration.
These bits are written by software to define the duration of the address setup phase in fmc_ker_ck cycles (refer to Figure 164 to Figure 176), used in asynchronous accesses:
- 0000: ADDSET phase duration = 0 × fmc_ker_ck clock cycle
- ... 1111: ADDSET phase duration = 15 × fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, the address setup phase is always 1 flash clock period duration. In muxed mode, the minimum ADDSET value is 1.
23.8 NAND flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND flash memories.

The NAND bank is configured through dedicated registers (Section 23.8.7). The programmable memory parameters include access timings (shown in Table 201) and ECC configuration.

### Table 201. Programmable NAND flash access parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Access mode</th>
<th>Unit</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory setup time</td>
<td>Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>1</td>
<td>255</td>
</tr>
<tr>
<td>Memory wait</td>
<td>Minimum duration (in fmc_ker_ck clock cycles) of the command assertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>2</td>
<td>255</td>
</tr>
<tr>
<td>Memory hold</td>
<td>Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command deassertion</td>
<td>Read/Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>1</td>
<td>254</td>
</tr>
<tr>
<td>Memory databus high-Z</td>
<td>Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started</td>
<td>Write</td>
<td>AHB clock cycle (fmc_ker_ck)</td>
<td>0</td>
<td>254</td>
</tr>
</tbody>
</table>

23.8.1 External memory interface signals

The following tables list the signals that are typically used to interface NAND flash memories.

*Note: The prefix "N" identifies the signals which are active low.*

### 8-bit NAND flash memory

#### Table 202. 8-bit NAND flash memory

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17]</td>
<td>O</td>
<td>NAND flash address latch enable (ALE) signal</td>
</tr>
<tr>
<td>A[16]</td>
<td>O</td>
<td>NAND flash command latch enable (CLE) signal</td>
</tr>
<tr>
<td>D[7:0]</td>
<td>I/O</td>
<td>8-bit multiplexed, bidirectional address/data bus</td>
</tr>
<tr>
<td>NCE</td>
<td>O</td>
<td>Chip Select</td>
</tr>
<tr>
<td>NOE(= NRE)</td>
<td>O</td>
<td>Output enable (memory signal name: read enable, NRE)</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NWAIT/INT</td>
<td>I</td>
<td>NAND flash ready/busy input signal to the FMC</td>
</tr>
</tbody>
</table>
Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

### 16-bit NAND flash memory

#### Table 203. 16-bit NAND flash memory

<table>
<thead>
<tr>
<th>FMC pin name</th>
<th>I/O</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>A[17]</td>
<td>O</td>
<td>NAND flash address latch enable (ALE) signal</td>
</tr>
<tr>
<td>A[16]</td>
<td>O</td>
<td>NAND flash command latch enable (CLE) signal</td>
</tr>
<tr>
<td>D[15:0]</td>
<td>I/O</td>
<td>16-bit multiplexed, bidirectional address/data bus</td>
</tr>
<tr>
<td>NCE</td>
<td>O</td>
<td>Chip Select</td>
</tr>
<tr>
<td>NOE (= NRE)</td>
<td>O</td>
<td>Output enable (memory signal name: read enable, NRE)</td>
</tr>
<tr>
<td>NWE</td>
<td>O</td>
<td>Write enable</td>
</tr>
<tr>
<td>NWAIT/INT</td>
<td>I</td>
<td>NAND flash ready/busy input signal to the FMC</td>
</tr>
</tbody>
</table>

**Note:** Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

#### 23.8.2 NAND flash supported memories and transactions

*Table 204 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.*

#### Table 204. Supported memories and transactions

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND 8-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>8</td>
<td>Y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>8</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 8 FMC accesses</td>
</tr>
<tr>
<td></td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>8</td>
<td>Y</td>
<td>Split into 8 FMC accesses</td>
</tr>
</tbody>
</table>
The NAND flash memory bank is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_ECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the three phases of any NAND flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. Figure 182 shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

### Table 204. Supported memories and transactions (continued)

<table>
<thead>
<tr>
<th>Device</th>
<th>Mode</th>
<th>R/W</th>
<th>AXI data size</th>
<th>Memory data size</th>
<th>Allowed/not allowed</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>8</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>8</td>
<td>16</td>
<td>N</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>16</td>
<td>16</td>
<td>Y</td>
<td>-</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 2 FMC accesses</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>R</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
<tr>
<td>NAND 16-bit</td>
<td>Asynchronous</td>
<td>W</td>
<td>32</td>
<td>16</td>
<td>Y</td>
<td>Split into 4 FMC accesses</td>
</tr>
</tbody>
</table>
23.8.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the corresponding memory bank by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see Section 23.8.5: NAND flash prewait feature) registers according to the characteristics of the NAND flash memory (PWID bits for the data bus width of the NAND flash memory, PWAITEN = 0 or 1 as needed, see Section 23.6.2: NAND flash memory address mapping for timing configuration).

2. The CPU performs a byte write to the common memory space, with data byte equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The LE input of the NAND flash memory is active during the write strobe (low pulse on NWE), thus the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.

3. The CPU can send the start address (STARTAD) for a read operation by writing four bytes (or three for smaller capacity devices), STARTAD[7:0], STARTAD[16:9], STARTAD[24:17] and finally STARTAD[25] (for 64 Mb x 8 bit NAND flash memories) in the common memory or attribute space. The ALE input of the NAND flash device is active during the write strobe (low pulse on NWE), thus the written bytes are interpreted as the start address for read operations. Using the attribute memory space makes it possible to use a different timing configuration of the FMC, which can be used...
to implement the prewait functionality needed by some NAND flash memories (see details in Section 23.8.5: NAND flash prewait feature).

4. The controller waits for the NAND flash memory to be ready (R/NB signal high), before starting a new access to the same or another memory bank. While waiting, the controller holds the NCE signal active (low).

5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.

6. The next NAND flash page can be read without any CPU command or address write operation. This can be done in three different ways:
   – by simply performing the operation described in step 5
   – a new random address can be accessed by restarting the operation at step 3
   – a new command can be sent to the NAND flash device by restarting at step 2

### 23.8.5 NAND flash prewait feature

Some NAND flash devices require that, after writing the last part of the address, the controller waits for the R/NB signal to go low. (see Figure 183).

**Figure 183. Access to non ‘CE don’t care’ NAND-flash**

1. CPU wrote byte 0x00 at address 0x7001 0000.
2. CPU wrote byte A7–A0 at address 0x7002 0000.
3. CPU wrote byte A16–A9 at address 0x7002 0000.
4. CPU wrote byte A24–A17 at address 0x7002 0000.
5. CPU wrote byte A25 at address 0x8802 0000: FMC performs a write access using FMC_PATT2 timing definition, where ATTHOLD ≥ 7 (providing that (7+1) × fmc_ker_ck = 112 ns > tWB max). This guarantees that NCE remains low until R/NB goes low and high again (only requested for NAND flash memories where NCE is not don’t care).
When this function is required, it can be performed by programming the MEMHOLD value to meet the t\textsubscript{WB} timing. However, any CPU read access to NAND flash memory has a hold delay of (MEMHOLD + 1) fmc\textsubscript{ker} ck cycles, and any CPU write access has a hold delay of (MEMHOLD) fmc\textsubscript{ker} ck cycles that is inserted between the rising edge of the NWE signal and the next access.

To cope with this timing constraint, the attribute memory space can be used by programming its timing register with an ATTHOLD value that meets the t\textsubscript{WB} timing, and by keeping the MEMHOLD value at its minimum value. The CPU must then use the common memory space for all NAND flash read and write accesses, except when writing the last address byte to the NAND flash device, where the CPU must write to the attribute memory space.

### 23.8.6 Computation of the error correction code (ECC) in NAND flash memory

The FMC includes an error correction code computation hardware block. It reduces the host CPU workload when processing the ECC by software. The ECC block is associated with NAND bank.

The ECC algorithm implemented in the FMC can perform 1-bit error correction and 2-bit error detection per 256, 512, 1 024, 2 048, 4 096 or 8 192 bytes read or written from/to the NAND flash memory. It is based on the Hamming coding algorithm and consists in calculating the row and column parity.

The ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory bank is active.

The ECC operates as follows:
- When accessing NAND flash bank, the data present on the D\textsubscript{15:0} bus is latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU, the FMC\_ECCR registers must be read to retrieve the computed value. Once read, they should be cleared by resetting the ECCEN bit to ‘0’. To compute a new data block, the ECCEN bit must be set to one in the FMC\_PCR registers.

Execute below the sequence to perform an ECC computation:
1. Enable the ECCEN bit in the FMC\_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC block computes the ECC value.
3. Wait until the ECC code is ready (FIFO empty).
4. Read the ECC value available in the FMC\_ECCR register and store it in a variable.
5. Clear the ECCEN bit and then enable it in the FMC\_PCR register before reading back the written data from the NAND page. While the NAND page is read, the ECC block computes the ECC value.
6. Read the new ECC value available in the FMC\_ECCR register.
7. If the two ECC values are the same, no correction is required, otherwise there is an ECC error and the software correction routine returns information on whether the error can be corrected or not.
### NAND flash controller registers

**NAND flash control registers (FMC_PCR)**

Address offset: 0x80  
Reset value: 0x0000 0018

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 19:17 | ECCPS[2:0]: ECC page size.  
These bits define the page size for the extended ECC:  
000: 256 bytes  
001: 512 bytes  
010: 1024 bytes  
011: 2048 bytes  
100: 4096 bytes  
101: 8192 bytes |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 16:13 | TAR[3:0]: ALE to RE delay.  
These bits set time from ALE low to RE low in number of fmc_ker_ck clock cycles.  
Time is: \( t_{ar} = (TAR + SET + 2) \times t_{fmc\_ker\_ck} \) \( t_{fmc\_ker\_ck} \) is the FMC clock period  
0000: 1 \( fmc\_ker\_ck \) cycle (default)  
1111: 16 \( fmc\_ker\_ck \) cycles  
Note: Set is MEMSET or ATTSET according to the addressed space. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 12:9  | TCLR[3:0]: CLE to RE delay.  
These bits set time from CLE low to RE low in number of fmc_ker_ck clock cycles. The time is give by the following formula:  
\[ t_{clr} = (TCLR + SET + 2) \times t_{fmc\_ker\_ck} \] \( t_{fmc\_ker\_ck} \) is the fmc_ker_ck clock period  
0000: 1 \( fmc\_ker\_ck \) cycle (default)  
1111: 16 \( fmc\_ker\_ck \) cycles  
Note: Set is MEMSET or ATTSET according to the addressed space. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 8:7   | Reserved, must be kept at reset value. |
|-----------|------|------|------|------|------|------|------|------|------|-----------|------|
|           |      |      |      |      |      |      |      |      |      |           |      |

| Bit 6     | ECCEN: ECC computation logic enable bit  
0: ECC logic is disabled and reset (default after reset),  
1: ECC logic is enabled. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 5:4   | PWID[1:0]: Data bus width.  
These bits define the external memory device width.  
00: 8 bits  
01: 16 bits (default after reset).  
10: reserved.  
11: reserved. |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:17  **ECCPS[2:0]: ECC page size.**

These bits define the page size for the extended ECC:

- 000: 256 bytes
- 001: 512 bytes
- 010: 1024 bytes
- 011: 2048 bytes
- 100: 4096 bytes
- 101: 8192 bytes

Bits 16:13  **TAR[3:0]: ALE to RE delay.**

These bits set time from ALE low to RE low in number of fmc_ker_ck clock cycles.

Time is: \( t_{ar} = (TAR + SET + 2) \times t_{fmc\_ker\_ck} \) \( t_{fmc\_ker\_ck} \) is the FMC clock period

- 0000: 1 \( fmc\_ker\_ck \) cycle (default)
- 1111: 16 \( fmc\_ker\_ck \) cycles

Note: Set is MEMSET or ATTSET according to the addressed space.

Bits 12:9  **TCLR[3:0]: CLE to RE delay.**

These bits set time from CLE low to RE low in number of fmc_ker_ck clock cycles. The time is given by the following formula:

\[ t_{clr} = (TCLR + SET + 2) \times t_{fmc\_ker\_ck} \] \( t_{fmc\_ker\_ck} \) is the fmc_ker_ck clock period

- 0000: 1 \( fmc\_ker\_ck \) cycle (default)
- 1111: 16 \( fmc\_ker\_ck \) cycles

Note: Set is MEMSET or ATTSET according to the addressed space.

Bits 8:7  Reserved, must be kept at reset value.

Bit 6  **ECCEN: ECC computation logic enable bit**

- 0: ECC logic is disabled and reset (default after reset),
- 1: ECC logic is enabled.

Bits 5:4  **PWID[1:0]: Data bus width.**

These bits define the external memory device width.

- 00: 8 bits
- 01: 16 bits (default after reset).
- 10: reserved.
- 11: reserved.
FIFO status and interrupt register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0040

This register contains information about the FIFO status and interrupt. The FMC features a FIFO that is used when writing to memories to transfer up to 16 words of data.

This is used to quickly write to the FIFO and free the AXI bus for transactions to peripherals other than the FMC, while the FMC is draining its FIFO into the memory. One of these register bits indicates the status of the FIFO, for ECC purposes.

The ECC is calculated while the data are written to the memory. To read the correct ECC, the software must consequently wait until the FIFO is empty.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| 31:7 | Reserved, must be kept at reset value. |

Bit 6 **FEMPT**: FIFO empty.

Read-only bit that provides the status of the FIFO

0: FIFO not empty

1: FIFO empty

Bit 5 **IFEN**: Interrupt falling edge detection enable bit

0: Interrupt falling edge detection request disabled

1: Interrupt falling edge detection request enabled

Bit 4 **ILEN**: Interrupt high-level detection enable bit

0: Interrupt high-level detection request disabled

1: Interrupt high-level detection request enabled

Bit 3 **IREN**: Interrupt rising edge detection enable bit

0: Interrupt rising edge detection request disabled

1: Interrupt rising edge detection request enabled
Bit 2 **IFS:** Interrupt falling edge status
The flag is set by hardware and reset by software.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

*Note:* If this bit is written by software to 1 it is set.

Bit 1 **ILS:** Interrupt high-level status
The flag is set by hardware and reset by software.
0: No interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IRS:** Interrupt rising edge status
The flag is set by hardware and reset by software.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

*Note:* If this bit is written by software to 1 it is set.

**Common memory space timing register (FMC_PMEM)**

Address offset: 0x88

Reset value: 0xFCFC FCFC

The FMC_PMEM read/write register contains the timing information for NAND flash memory bank. This information is used to access either the common memory space of the NAND flash for command, address write access and data read/write access.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMHI[7:0]</td>
<td>MEMHOLD[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>MEMHIZ[7:0]: Common memory x data bus Hi-Z time</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept Hi-Z after the start of a NAND flash write access to common memory space. This is only valid for write transactions: 0000 0000: 0 x fmc_ker_ck cycle 1111 1110: 254 x fmc_ker_ck cycles 1111 1111: reserved.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 23:16</th>
<th>MEMHOLD[7:0]: Common memory hold time</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits define the number of fmc_ker_ck clock cycles for write accesses and fmc_ker_ck+1 clock cycles for read accesses during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), for NAND flash read or write access to common memory space: 0000 0000: reserved. 0000 0001: 1 fmc_ker_ck cycle for write access / 3 fmc_ker_ck cycle for read access 1111 1110: 254 fmc_ker_ck cycles for write access / 257 fmc_ker_ck cycles for read access 1111 1111: reserved.</td>
<td></td>
</tr>
</tbody>
</table>
Attribute memory space timing registers (FMC_PATT)

Address offset: 0x8C

Reset value: 0xFCFC FCFC

The FMC_PATT read/write register contains the timing information for NAND flash memory bank. It is used for 8-bit accesses to the attribute memory space of the NAND flash for the last address write access if the timing must differ from that of previous accesses (for Ready/Busy management, refer to Section 23.8.5: NAND flash prewait feature).

### Bits 15:8 MEMWAIT[7:0]: Common memory wait time
These bits define the minimum number of fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:

- 0000 0000: reserved
- 0000 0001: x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
- 1111 1110: 255 x fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)
- 1111 1111: reserved.

### Bits 7:0 MEMSET[7:0]: Common memory x setup time
These bits define the number of fmc_ker_ck clock cycles to set up the address before the command assertion (NWE, NOE), for NAND flash read or write access to common memory space:

- 0000 0000: fmc_ker_ck cycles
- 1111 1110: 255 x fmc_ker_ck cycles
- 1111 1111: reserved

### Bits 31:24 ATTHIZ[7:0]: Attribute memory data bus Hi-Z time
These bits define the number of fmc_ker_ck clock cycles during which the data bus is kept in Hi-Z after the start of a NAND flash write access to attribute memory space on socket. Only valid for write transaction:

- 0000 0000: 0 x fmc_ker_ck cycle
- 1111 1110: 254 x fmc_ker_ck cycles
- 1111 1111: reserved.

### Bits 23:16 ATTHOLD[7:0]: Attribute memory hold time
These bits define the number of fmc_ker_ck clock cycles during which the address is held (and data for write access) after the command deassertion (NWE, NOE), for NAND flash read or write access to attribute memory space:

- 0000 0000: reserved
- 0000 0001: 1 x fmc_ker_ck cycle
- 1111 1110: 254 x fmc_ker_ck cycles
- 1111 1111: reserved.
Bits 15:8 **ATTWAIT[7:0]**: Attribute memory wait time

These bits define the minimum number of \( \times \text{fmc}_\text{ker}_\text{ck} \) (+1) clock cycles to assert the command (NWE, NOE), for NAND flash read or write access to attribute memory space. The duration for command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of \( \text{fmc}_\text{ker}_\text{ck} \):

- 0000 0000: reserved
- 0000 0001: 2 \( \times \text{fmc}_\text{ker}_\text{ck} \) cycles (+ wait cycle introduced by deassertion of NWAIT)
- 1111 1110: 255 \( \times \text{fmc}_\text{ker}_\text{ck} \) cycles (+ wait cycle introduced by deasserting NWAIT)
- 1111 1111: reserved.

Bits 7:0 **ATTSET[7:0]**: Attribute memory setup time

These bits define the number of \( \times \text{fmc}_\text{ker}_\text{ck} \) (+1) clock cycles to set up address before the command assertion (NWE, NOE), for NAND flash read or write access to attribute memory space:

- 0000 0000: 1 \( \times \text{fmc}_\text{ker}_\text{ck} \) cycle
- 1111 1110: 255 \( \times \text{fmc}_\text{ker}_\text{ck} \) cycles
- 1111 1111: reserved.
ECC result registers (FMC_ECCR)

Address offset: 0x94
Reset value: 0x0000 0000

This register contain the current error correction code value computed by the ECC computation modules of the FMC NAND controller. When the CPU reads/writes the data from a NAND flash memory page at the correct address (refer to Section 23.8.6: Computation of the error correction code (ECC) in NAND flash memory), the data read/written from/to the NAND flash memory are processed automatically by the ECC computation module. When X bytes have been read (according to the ECCPS field in the FMC_PCR registers), the CPU must read the computed ECC value from the FMC_ECC registers. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and, to correct it otherwise. The FMC_ECC register should be cleared after being read by setting the ECCEN bit to '0'. To compute a new data block, the ECCEN bit must be set to '1'.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECC[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 ECC[31:0]: ECC result
This field contains the value computed by the ECC computation logic. Table 205 describes the contents of these bitfields.

Table 205. ECC result relevant bits

<table>
<thead>
<tr>
<th>ECCPS[2:0]</th>
<th>Page size in bytes</th>
<th>ECC bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>256</td>
<td>ECC[21:0]</td>
</tr>
<tr>
<td>001</td>
<td>512</td>
<td>ECC[23:0]</td>
</tr>
<tr>
<td>010</td>
<td>1024</td>
<td>ECC[25:0]</td>
</tr>
<tr>
<td>011</td>
<td>2048</td>
<td>ECC[27:0]</td>
</tr>
<tr>
<td>100</td>
<td>4096</td>
<td>ECC[29:0]</td>
</tr>
<tr>
<td>101</td>
<td>8192</td>
<td>ECC[31:0]</td>
</tr>
</tbody>
</table>
23.9 SDRAM controller

23.9.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM banks with independent configuration
- 8-bit, 16-bit, 32-bit data bus width
- 13-bits Address Row, 11-bits Address Column, 4 internal banks: 4x16Mx32bit (256 MB), 4x16Mx16bit (128 MB), 4x16Mx8bit (64 MB)
- Word, half-word, byte access
- SDRAM clock can be fmc_ker_ck/2 or fmc_ker_ck/3
- Automatic row and bank boundary management
- Multibank ping-pong access
- Programmable timing parameters
- Automatic Refresh operation with programmable Refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3
- Cacheable Read FIFO with depth of 6 lines x32-bit (6 x14-bit address tag)

23.9.2 SDRAM External memory interface signals

At startup, the SDRAM I/O pins used to interface the FMC SDRAM controller with the external SDRAM devices must configured by the user application. The SDRAM controller I/O pins which are not used by the application, can be used for other purposes.

<table>
<thead>
<tr>
<th>SDRAM signal</th>
<th>I/O type</th>
<th>Description</th>
<th>Alternate function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDCLK</td>
<td>O</td>
<td>SDRAM clock</td>
<td>-</td>
</tr>
</tbody>
</table>
| SDCKE[1:0]   | O        | SDCKE0: SDRAM Bank 1 Clock Enable  
SDCKE1: SDRAM Bank 2 Clock Enable | - |
| SDNE[1:0]    | O        | SDNE0: SDRAM Bank 1 Chip Enable  
SDNE1: SDRAM Bank 2 Chip Enable | - |
| D[31:0]      | I/O      | Bidirectional data bus | FMC_D[31:0] |
| BA[1:0]      | O        | Bank Address | FMC_A[15:14] |
| NRAS         | O        | Row Address Strobe | - |
| NCAS         | O        | Column Address Strobe | - |
| SDNWE        | O        | Write Enable | - |
| NBL[3:0]     | O        | Output Byte Mask for write accesses (memory signal name: DQM[3:0]) | FMC_NBL[3:0] |
23.9.3 SDRAM controller functional description

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (FMC_SDCLK).

SDRAM initialization

The initialization sequence is managed by software. If the two banks are used, the initialization sequence must be generated simultaneously to Bank 1 and Bank 2 by setting the Target Bank bits CTB1 and CTB2 in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register. The SDRAM clock frequency, RBURST and RPIPE must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTRx register. The TRP and TRC timings must be programmed in the FMC_SDTR1 register.
3. Set MODE bits to ‘001’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to start delivering the clock to the memory (SDCKE is driven high).
4. Wait during the prescribed delay period. Typical delay is around 100 μs (refer to the SDRAM datasheet for the required delay after power-up).
5. Set MODE bits to ‘010’ and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Precharge All” command.
6. Set MODE bits to ‘011’, and configure the Target Bank bits (CTB1 and/or CTB2) as well as the number of consecutive Auto-refresh commands (NRFS) in the FMC_SDCMR register. Refer to the SDRAM datasheet for the number of Auto-refresh commands that should be issued. Typical number is 8.
7. Configure the MRD field, set the MODE bits to ‘100’, and configure the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register to issue a “Load Mode Register” command and program the SDRAM device. In particular the Burst Length (BL) has to be set to ‘1’) and the CAS latency has to be selected. If the Mode Register is not the same for both SDRAM banks, this step has to be repeated twice, once for each bank and the Target Bank bits set accordingly. For mobile SDRAM devices, the MRD field is also used to configure the extended mode register while issuing the Load Mode Register.
8. Program the refresh rate in the FMC_SDRTR register.

The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted to SDRAM devices.

At this stage the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device. Therefore the SDRAM device must be first reinitialized after reset before issuing any new access by the NOR flash/PSRAM/SRAM or NAND flash controller.

Note: If two SDRAM devices are connected to the FMC, all the accesses performed at the same time to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for SDRAM Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.
SDRAM controller write cycle

The SDRAM controller accepts single and burst write requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row for each bank to be able to perform consecutive write accesses to different banks (Multibank ping-pong access).

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

The SDRAM controller always checks the next access.
- If the next access is in the same row or in another active row, the write operation is carried out,
- if the next access targets another row (not active), the SDRAM controller generates a precharge command, activates the new row and initiates a write command.
SDRAM controller read cycle

The SDRAM controller accepts single and burst read requests and translates them into single memory accesses. In both cases, the SDRAM controller keeps track of the active row in each bank to be able to perform consecutive read accesses in different banks (Multibank ping-pong access).

The FMC SDRAM controller features a Cacheable read FIFO (6 lines x 32 bits). It is used to store data read in advance during the CAS latency period (up to 3 memory clock cycles, programmed FMC_SDCRx) and during the RPIPE delay when set to 2xfmc_ker_ck clock cycles as configured in FMC_SDCR1) following this formula: CAS Latency + 1 + (RPIPE DIV2). The RBURST bit must be set in the FMC_SDCR1 register to anticipate the next read access.

Examples:
- CAS=3, RPIPE= 2xfmc_ker_ck. In this case, 5 data (not committed) are stored in the FIFO (4 data during CAS latency and 1 data during RPIPE delay)
- CAS=3, RPIPE= 1xfmc_ker_ck. In this case, 4 data (not committed) are stored in the FIFO (4 data during CAS latency)

The read FIFO features a 14-bit address tag to each line to identify its content: 11 bits for the column address, 2 bits to select the internal bank and the active row, and 1 bit to select the SDRAM device.

When the end of the row is reached in advance during an burst read transaction, the data read in advance (not committed) are not stored in the read FIFO. For single read access, data are correctly stored in the FIFO.
Each time a read request occurs, the SDRAM controller checks:

- If the address matches one of the address tags, data are directly read from the FIFO and the corresponding address tag/line content is cleared and the remaining data in the FIFO are compacted to avoid empty lines.
- Otherwise, a new read command is issued to the memory and the FIFO is updated with new data. If the FIFO is full, the older data are lost.

**Figure 186. Logic diagram of Read access with RBURST bit set (CAS=2, RPIPE=0)**

During a write access or a Precharge command, the read FIFO is flushed and ready to be filled with new data.

After the first read request, if the current access was not performed to a row boundary, the SDRAM controller anticipates the next read access during the CAS latency period and the RPIPE delay (if configured). This is done by incrementing the memory address. The following condition must be met:

- RBURST control bit should be set to ‘1’ in the FMC_SDCR1 register.
The address management depends on the next AXI request:

- Next request is sequential (Burst access)
  
  In this case, the SDRAM controller increments the address.

- Next request is not sequential
  
  - If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  
  - If the new read request does not target an active row, the SDRAM controller generates a Precharge command, activates the new row, and initiates a read command.

If the RBURST is reset, the read FIFO is not used.

**Row and bank boundary management**

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row,
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Active commands to match TRP parameter (only if the next access is in a different row in the same bank),
- Between Active and Read commands to match the TRCD parameter.

These parameters are defined into the FMC_SDTRx register.

Refer to *Figure 184* and *Figure 185* for read and burst write access crossing a row boundary.
Figure 187. Read access crossing row boundary

Figure 188. Write access crossing row boundary
If the next access is sequential and the current access crosses a bank boundary, the SDRAM controller activates the first row in the next bank and initiates a new read/write command. Two cases are possible:

- If the current bank is not the last one, the active row in the new bank must be precharged. At a bank boundary, the automatic activation of the next row is supported for all rows/columns and data bus width configuration.
- If the current bank is the last one, the automatic activation of the next row is supported only when addressing 13-bit rows, 11-bit columns, 4 internal banks and 32-bit data bus SDRAM devices. Otherwise, the SDRAM address range is violated and an AXI slave error is generated.
- In case of 13-bit row address, 11-bit column address, 4 internal banks and bus width 32-bit SDRAM memories, at boundary bank, the SDRAM controller continues to read/write from the second SDRAM device (assuming it has been initialized):
  a) The SDRAM controller activates the first row (after precharging the active row, if there is already an active row in the first internal bank, and initiates a new read/write command.
  b) If the first row is already activated, the SDRAM controller just initiates a read/write command.

SDRAM controller refresh cycle

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues auto-refresh commands. An internal counter is loaded with the COUNT value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, an internal pulse is generated.

If a memory access is ongoing, the auto-refresh request is delayed. However, if the memory access and the auto-refresh requests are generated simultaneously, the auto-refresh request takes precedence.

If the memory access occurs during an auto-refresh operation, the request is buffered and processed when the auto-refresh is complete.

If a new auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = ‘1’).

If SDRAM lines are not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge ALL) command before the auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR Command Mode register (Mode bits = ‘011’), a PALL command (Mode bits = ‘010’) must be issued first.
23.9.4 Low-power modes

Two low-power modes are available:

- **Self-refresh mode**
  The auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.

- **Power-down mode**
  The auto-refresh cycles are performed by the SDRAM controller.

**Self-refresh mode**

This mode is selected by setting the MODE bits to ‘101’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

The SDRAM clock stops running after a TRAS delay and the internal refresh timer stops counting only if one of the following conditions is met:

- A Self-refresh command is issued to both devices
- One of the devices is not activated (SDRAM bank is not initialized).

Before entering Self-Refresh mode, the SDRAM controller automatically issues a PALL command.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Self-refresh mode and the BUSY status flag remains set.

In Self-refresh mode, all SDRAM device inputs become don’t care except for SDCKE which remains low.

The SDRAM device must remain in Self-refresh mode for a minimum period of time of TRAS and can remain in Self-refresh mode for an indefinite period beyond that. To guarantee this minimum period, the BUSY status flag remains high after the Self-refresh activation during a TRAS delay.

As soon as an SDRAM device is selected, the SDRAM controller generates a sequence of commands to exit from Self-refresh mode. After the memory access, the selected device remains in Normal mode.

To exit from Self-refresh, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.
Figure 189. Self-refresh mode

- **SDCLK**
- **SDCKE**
- **COMMAND**:
  - PRECHARGE
  - NOP
  - AUTO REFRESH
  - NOP or COMMAND INHERIT
  - AUTO REFRESH
- **DOM/DOML/DOMU**
- **A0- A9 A11, A12**
- **A10**:
  - ALL BANKS
- **Data[31:0]**
  - Hi-Z

\[tRAS(\text{min})\]

Events:
- Precharge all active banks
- Enter Self-refresh mode
- Exit Self-refresh mode
- CLK stable prior to existing Self-refresh mode
- (restart refresh timebase)
Power-down mode

This mode is selected by setting the MODE bits to ‘110’ and by configuring the Target Bank bits (CTB1 and/or CTB2) in the FMC_SDCMR register.

If the Write data FIFO is not empty, all data are sent to the memory before activating the Power-down mode.

As soon as an SDRAM device is selected, the SDRAM controller exits from the Power-down mode. After the memory access, the selected SDRAM device remains in Normal mode.

During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

To exit from Power-down mode, the MODE bits must be set to ‘000’ (Normal mode) and the Target Bank bits (CTB1 and/or CTB2) must be configured in the FMC_SDCMR register.
23.9.5 SDRAM controller registers

SDRAM Control registers for SDRAM memory bank x (FMC_SDCRx)

Address offset: 0x140+ 0x4* (x − 1), (x = 1 to 2)

Reset value: 0x0000 02D0

This register contains the control parameters for each SDRAM memory bank

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:13 **RPIPE[1:0]**: Read pipe

These bits define the delay, in fmc_ker_ck clock cycles, for reading data after CAS latency.

00: No fmc_ker_ck clock cycle delay
01: One fmc_ker_ck clock cycle delay
10: Two fmc_ker_ck clock cycle delay
11: reserved.

*Note: The corresponding bits in the FMC_SDCR2 register is read only.*

Bit 12 **RBURST**: Burst read

This bit enables Burst read mode. The SDRAM controller anticipates the next read commands during the CAS latency and stores data in the Read FIFO.

0: single read requests are not managed as bursts
1: single read requests are always managed as bursts

*Note: The corresponding bit in the FMC_SDCR2 register is read only.*

Bits 11:10 **SDCLK[1:0]**: SDRAM clock configuration

These bits define the SDRAM clock period for both SDRAM banks and allow disabling the clock before changing the frequency. In this case the SDRAM must be re-initialized.

00: SDCLK clock disabled
01: Reserved
10: SDCLK period = 2 x fmc_ker_ck periods
11: SDCLK period = 3 x fmc_ker_ck periods

*Note: The corresponding bits in the FMC_SDCR2 register is read only.*

Bit 9 **WP**: Write protection

This bit enables Write mode access to the SDRAM bank.

0: Write accesses allowed
1: Write accesses ignored
Note: Before modifying the RBURST or RPIPE settings or disabling the SDCLK clock, the user must first send a PALL command to make sure ongoing operations are complete.

SDRAM Timing registers for SDRAM memory bank x (FMC_SDTRx)

Address offset: 0x148 + 0x4 * (x - 1), (x = 1 to 2)

Reset value: 0x0FFF FFFF

This register contains the timing parameters of each SDRAM bank

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>TRCD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>TRP[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>TWR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>TRC[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>TRAS[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>TXSR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>TMRD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>TRCD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>TRP[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>TWR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>TRC[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>TRAS[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>TXSR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>TMRD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>TRCD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>TRP[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>TWR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>TRC[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>TRAS[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>TXSR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>TMRD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>TRCD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>TRP[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>TWR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>TRC[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>TRAS[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>TXSR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>TMRD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>TRCD[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>TRP[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>TWR[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>TRC[3:0]</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 8:7  **CAS[1:0]:** CAS Latency
This bits sets the SDRAM CAS latency in number of memory clock cycles
00: reserved.
01: 1 cycle
10: 2 cycles
11: 3 cycles

Bit 6  **NB:** Number of internal banks
This bit sets the number of internal banks.
0: Two internal Banks
1: Four internal Banks

Bits 5:4  **MWID[1:0]:** Memory data bus width.
These bits define the memory device width.
00: 8 bits
01: 16 bits
10: 32 bits
11: reserved.

Bits 3:2  **NR[1:0]:** Number of row address bits
These bits define the number of bits of a row address.
00: 11 bit
01: 12 bits
10: 13 bits
11: reserved.

Bits 1:0  **NC[1:0]:** Number of column address bits
These bits define the number of bits of a column address.
00: 8 bits
01: 9 bits
10: 10 bits
11: 11 bits.
Bits 31:28  Reserved, must be kept at reset value.

Bits 27:24  TRCD[3:0]: Row to column delay
These bits define the delay between the Activate command and a Read/Write command in number of memory clock cycles.
0000: 1 cycle.
0001: 2 cycles
....
1111: 16 cycles

Bits 23:20  TRP[3:0]: Row precharge delay
These bits define the delay between a Precharge command and another command in number of memory clock cycles. The TRP timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP must be programmed with the timing of the slowest device.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles
Note: The corresponding bits in the FMC_SDTR2 register are don’t care.

Bits 19:16  TWR[3:0]: Recovery delay
These bits define the delay between a Write and a Precharge command in number of memory clock cycles.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles
Note: TWR must be programmed to match the write recovery time (tWR) defined in the SDRAM datasheet, and to guarantee that:
TWR ≥ TRAS - TRCD and TWR ≥ TRC - TRCD - TRP
Example: TRAS = 4 cycles, TRCD = 2 cycles. So, TWR ≥ 2 cycles. TWR must be programmed to 0x1.
If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TWR timing corresponding to the slowest SDRAM device.
If only one SDRAM device is used, the TWR timing must be kept at reset value (0xF) for the not used bank.

Bits 15:12  TRC[3:0]: Row cycle delay
These bits define the delay between the Refresh command and the Activate command, as well as the delay between two consecutive Refresh commands. It is expressed in number of memory clock cycles. The TRC timing is only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRC must be programmed with the timings of the slowest device.
0000: 1 cycle
0001: 2 cycles
....
1111: 16 cycles
Note: TRC must match the TRC and TRFC (Auto Refresh period) timings defined in the SDRAM device datasheet.
Note: The corresponding bits in the FMC_SDTR2 register are don’t care.
Bits 11:8 **TRAS[3:0]:** Self refresh time
These bits define the minimum Self-refresh period in number of memory clock cycles.
0000: 1 cycle  
0001: 2 cycles  
....  
1111: 16 cycles  

Bits 7:4 **TXSR[3:0]:** Exit Self-refresh delay
These bits define the delay from releasing the Self-refresh command to issuing the Activate command in number of memory clock cycles.
0000: 1 cycle  
0001: 2 cycles  
....  
1111: 16 cycles  

Note: If two SDRAM devices are used, the FMC_SDTR1 and FMC_SDTR2 must be programmed with the same TXSR timing corresponding to the slowest SDRAM device.

Bits 3:0 **TMRD[3:0]:** Load Mode Register to Active
These bits define the delay between a Load Mode Register command and an Active or Refresh command in number of memory clock cycles.
0000: 1 cycle  
0001: 2 cycles  
....  
1111: 16 cycles  

Note: If two SDRAM devices are connected, all the accesses performed simultaneously to both devices by the Command Mode register (Load Mode Register command) are issued using the timing parameters configured for Bank 1 (TMRD and TRAS timings) in the FMC_SDTR1 register.

The TRP and TRC timings are only configured in the FMC_SDTR1 register. If two SDRAM devices are used, the TRP and TRC timings must be programmed with the timings of the slowest device.

**SDRAM Command mode register (FMC_SDCMR)**
Address offset: 0x150  
Reset value: 0x0000 0000  

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE field is written, the command is issued only to one or to both SDRAM banks according to CTB1 and CTB2 command bits. This register is the same for both SDRAM banks.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rd</td>
<td>rd</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>rm</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:23 Reserved, must be kept at reset value.
Flexible memory controller (FMC) RM0477

SDRAM refresh timer register (FMC_SDRTR)

Address offset: 0x154
Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the refresh cycles by configuring the Refresh Timer Count value.

\[
\text{Refresh rate} = (\text{COUNT} + 1) \times \text{SDRAM clock frequency}
\]

\[
\text{COUNT} = (\text{SDRAM refresh period} / \text{Number of rows}) - 20
\]
Below an example of refresh rate calculation:

\[
\text{Refresh rate} = \frac{64 \text{ ms}}{(8196 \text{ rows})} = 7.81 \mu s
\]

where 64 ms is the SDRAM refresh period.

\[7.81 \mu s \times 60 \text{ MHz} = 468.6\]

The refresh rate must be increased by 20 SDRAM clock cycles (as in the above example) to obtain a safe margin if an internal refresh request occurs when a read request has been accepted. It corresponds to a COUNT value of '0000111000000' (448).

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a refresh pulse when zero is reached. The COUNT value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is '0', no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, this 13-bit COUNT field is reloaded into the counter.

If a memory access is in progress, the Auto-refresh request is delayed. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

This register is common to SDRAM bank 1 and bank 2.

### Register Description

<table>
<thead>
<tr>
<th>Field</th>
<th>REIE</th>
<th>COUNT[12:0]</th>
<th>CRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:15</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 14</td>
<td><strong>REIE</strong>: RES Interrupt Enable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: Interrupt is disabled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: An Interrupt is generated if RE = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 13:1</td>
<td><strong>COUNT[12:0]</strong>: Refresh Timer Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This 13-bit field defines the refresh rate of the SDRAM device. It is expressed in number of memory clock cycles. It must be set at least to 58 SDRAM clock cycles (0x34).</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Refresh rate = (COUNT + 1) \times SDRAM frequency clock</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>COUNT = (SDRAM refresh period / Number of rows) - 20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>CRE</strong>: Clear Refresh error flag</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit is used to clear the Refresh Error Flag (RE) in the Status Register.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: no effect</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: Refresh Error flag is cleared</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The programmed COUNT value must not be equal to the sum of the following timings: TWR+TRP+TRC+TRCD+4 memory clock cycles.
SDRAM Status register (FMC_SDSR)

Address offset: 0x158
Reset value: 0x0000 0000

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

Bits 4:3 **MODES2[1:0]**: Status Mode for Bank 2
These bits define the Status Mode of SDRAM Bank 2.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bits 2:1 **MODES1[1:0]**: Status Mode for Bank 1
These bits define the Status Mode of SDRAM Bank 1.
00: Normal Mode
01: Self-refresh mode
10: Power-down mode

Bit 0 **RE**: Refresh error flag
0: No refresh error has been detected
1: A refresh error has been detected
An interrupt is generated if REIE = 1 and RE = 1

23.9.6 **FMC register map**

The following table summarizes the FMC registers.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>FMC_BCR1</td>
<td>[bits 31:0]</td>
</tr>
<tr>
<td></td>
<td>FMCEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BMAP[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WFDIS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CPSIZE[2:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CYCLEN</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>FMC_BCR2</td>
<td>[bits 15:0]</td>
</tr>
<tr>
<td></td>
<td>FACCEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WBKEN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MWID</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MTYP[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MUXEN</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 00 00 00 00 00 01 10 00 10 10 01 0 1 0 1 0
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10</td>
<td>FMC_BCR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>FMC_BCR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x04</td>
<td>FMC_BTR1</td>
<td>ACCM_OD[1:0]</td>
<td>DATLAT[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x0C</td>
<td>FMC_BTR2</td>
<td>ACCM_OD[1:0]</td>
<td>DATLAT[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x14</td>
<td>FMC_BTR3</td>
<td>ACCM_OD[1:0]</td>
<td>DATLAT[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x1C</td>
<td>FMC_BTR4</td>
<td>ACCM_OD[1:0]</td>
<td>DATLAT[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x104</td>
<td>FMC_BWTR1</td>
<td>ACCM_OD[1:0]</td>
<td>BUSTURN[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x10C</td>
<td>FMC_BWTR2</td>
<td>ACCM_OD[1:0]</td>
<td>BUSTURN[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x114</td>
<td>FMC_BWTR3</td>
<td>ACCM_OD[1:0]</td>
<td>BUSTURN[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x11C</td>
<td>FMC_BWTR4</td>
<td>ACCM_OD[1:0]</td>
<td>BUSTURN[3:0]</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x80</td>
<td>FMC_PCR</td>
<td></td>
<td>ECCPS [2:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0x84</td>
<td>FMC_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x88</td>
<td>FMC_PMEM</td>
<td></td>
<td>MEMHZ[7:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x8C</td>
<td>FMC_PATT</td>
<td></td>
<td>ATHZ[7:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x94</td>
<td>FMC_ECCR</td>
<td></td>
<td>ECC[31:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x140</td>
<td>FMC_SDRCR</td>
<td></td>
<td>RPIPE[1:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 149 for the register boundary addresses.
24 Extended-SPI interface (XSPI)

24.1 Introduction

The XSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- **indirect mode**: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- **automatic status-polling mode**: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- **memory-mapped mode**: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The XSPI supports the following protocols with associated frame formats:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

24.2 XSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- Support for single, dual, quad, and octal communication
- Dual-memory configuration, where 8 bits can be sent/received simultaneously by accessing two quad or two octal memories in parallel
- XSPI mode accessing a single 16-bit memory
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AXI interface with transaction acceptance limited to one: the interface accepts the next transfer on AXI bus only once the previous is completed on memory side.
- Dual chip select support (NCS1 and NCS2)
## 24.3 XSPI implementation

**Table 208. XSPI implementation**

<table>
<thead>
<tr>
<th>Feature</th>
<th>XSPI1/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyperBus standard compliant</td>
<td>X</td>
</tr>
<tr>
<td>Xccela standard compliant</td>
<td>X</td>
</tr>
<tr>
<td>XSPI (JDES251C) standard compliant</td>
<td>X</td>
</tr>
<tr>
<td>AMBA® AXI compliant data interface</td>
<td>X</td>
</tr>
<tr>
<td>Asynchronous AHB clock versus kernel clock</td>
<td>X</td>
</tr>
<tr>
<td>Functional modes: indirect, automatic status-polling, and memory-mapped</td>
<td>X</td>
</tr>
<tr>
<td>Dual chip select support (NCS1 and NCS2)</td>
<td>X</td>
</tr>
<tr>
<td>Read and write support in memory-mapped mode</td>
<td>X</td>
</tr>
<tr>
<td>Dual-quad configuration</td>
<td>X</td>
</tr>
<tr>
<td>Dual-octal configuration</td>
<td>X</td>
</tr>
<tr>
<td>SDR (single-data rate) and DTR (double-transfer rate)</td>
<td>X</td>
</tr>
<tr>
<td>Data strobe (DS, DQS)</td>
<td>X</td>
</tr>
<tr>
<td>Fully programmable opcode</td>
<td>X</td>
</tr>
<tr>
<td>Fully programmable frame format</td>
<td>X</td>
</tr>
<tr>
<td>Integrated FIFO for reception and transmission</td>
<td>X</td>
</tr>
<tr>
<td>8, 16, and 32-bit data accesses</td>
<td>X</td>
</tr>
<tr>
<td>Interrupt on FIFO threshold, timeout, operation complete, and access error</td>
<td>X</td>
</tr>
<tr>
<td>Compliant with dual-XSPI arbiter (communication regulation)</td>
<td>X</td>
</tr>
<tr>
<td>Extended CSHT timeout</td>
<td>X</td>
</tr>
<tr>
<td>Memory-mapped write</td>
<td>X</td>
</tr>
<tr>
<td>Refresh counter</td>
<td>X</td>
</tr>
<tr>
<td>High-speed interface</td>
<td>X</td>
</tr>
<tr>
<td>GP/HPDMA interface</td>
<td>X</td>
</tr>
<tr>
<td>Prefetch disable</td>
<td>-</td>
</tr>
<tr>
<td>Prefetch hardware software</td>
<td>-</td>
</tr>
</tbody>
</table>
24.4 **XSPI functional description**

24.4.1 **XSPI block diagram**

The block diagrams provided below correspond to a single XSPI connected to the I/O ports. In case several XSPIs are connected to the I/O ports through the XSPIM I/O manager, refer to Section 25: XSPI I/O manager (XSPIM) for more details.

Figure 191. XSPI block diagram for 16-bit configuration

1. The configuration depends on the package (see the datasheet for more details).
1. The configuration depends on the package (see the datasheet for more details).
Figure 193. XSPI block diagram for octal configuration

(1) The Octo-SPI memory is connected to XSPIM_P1_IO[0:7] but it can also be connected to XSPIM_P1_IO[8:15].

(2) The configuration with NCS2 used corresponds to a semi-static management with two external memories hooked on the same XSPI bus. The software selects which memory is accessed.
Figure 194. XSPI block diagram in quad configuration

(1) The Quad-SPI memory is connected to XSPIM_P1_IO[0:3] but it can also be connected to XSPIM_P1_IO[4:7], XSPIM_P1_IO[8:11], or XSPIM_P1_IO[12:15].

(2) The configuration with NCS2 used corresponds to a semi-static management with two external memories hooked on the same XSPI bus. The software selects which memory is accessed.
24.4.2 XSPI pins and internal signals

Table 209. XSPI input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSPIM_Px_NCLK</td>
<td>Output</td>
<td>XSPI inverted clock to support 1.8 V HyperBus protocol.</td>
</tr>
<tr>
<td>XSPIM_Px_CLK</td>
<td>Output</td>
<td>XSPI clock</td>
</tr>
<tr>
<td>XSPIM_Px_IOx (n = 0 to 15)</td>
<td>Input/output</td>
<td>XSPI data pins</td>
</tr>
<tr>
<td>XSPIM_Px_NCS1,2</td>
<td>Output</td>
<td>Chip select for the memory</td>
</tr>
<tr>
<td>XSPIM_Px_DQS0,1</td>
<td>Input/output</td>
<td>Data strobe/write mask signal from/to the memory</td>
</tr>
</tbody>
</table>

Caution: Use the same configuration (output speed, HSLV) for all XSPI input/output pins to avoid any data corruption.

Table 210. XSPI internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xspl_hclk</td>
<td>Input</td>
<td>XSPI AHB clock</td>
</tr>
<tr>
<td>xspl_aclk</td>
<td>Input</td>
<td>XSPI AXI clock</td>
</tr>
</tbody>
</table>
24.4.3 Clock constraints

The XSPI AXI clock (xspi_aclk) has to be greater or equal than the XSPI kernel clock (xspi_ker_ck).

24.4.4 XSPI interface to memory modes

The XSPI supports the following protocols:
- regular-command protocol
- HyperBus protocol

The XSPI uses from 6 to 21 signals to interface with a memory, depending on the functional mode:
- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time).
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS0, DQS1: data strobe used only in regular-command protocol
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus used as possible remap for quad-SPI mode
- IO[15:8]:
  - data bus MSB used in dual-quad, dual-octal and 16-bit configurations
  - data bus used as possible remap for octal-SPI mode
  - IO[15:12] and IO[11:8] can also be used as possible remap for quad-SPI mode

24.4.5 XSPI regular-command protocol

When in regular-command protocol, the XSPI communicates with the external device using commands. Each command can include the following phases:
- instruction phase
- address phase
- alternate-byte phase
- dummy-cycle phase
- data phase

Only the data phase uses 16 bits. Instruction, address, and alternate phases use only the eight LSB of the bus as for octal configuration.
Any of these phases can be configured to be skipped, but single-phase commands supported are only those with instruction phase.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operation are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

**Figure 196. SDR read command in 16-bit configuration**

![Diagram showing the SDR read command in 16-bit configuration]

1. Data (such as D0, D1, D2) are sent in 16-bit configuration mode over IO[15:0]. Only the command and address are sent over IO[7:0] as for octal mode.

The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

**Instruction phase**

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured in ISIZE[1:0] of XSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of XSPI_IR.

The instruction phase can optionally send:
- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode, or in 16-bit SPI mode)

This can be configured using IMODE[2:0] of XSPI_CCR.

The instruction can be sent in DTR (double-transfer rate) mode on each rising and falling edge of the clock, by setting IDTR in XSPI_CCR.

When IMODE[2:0] = 000 in XSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

When in memory-mapped mode, the instruction used for the write operation is specified in XSPI_WIR and the instruction format is specified in XSPI_WCCR. The instruction used for the read operation and the instruction format are specified in XSPI_IR and XSPI_CCR.

**Address phase**

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured in ADSIZE[1:0] of XSPI_CCR.
In indirect and automatic status-polling modes, the address bytes to be sent are specified in ADDRESS[31:0] of XSPI_AR. In memory-mapped mode, the address is given directly via the AXI (from any master in the system).

The address phase can send:
- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode, or in 16-bit SPI mode)

This can be configured using ADMODE[2:0] of XSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in XSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in XSPI_WCCR. The address format for the read operation is specified in XSPI_CCR.

---

**Warning:** Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiple the address by two when accessing the memory registers.

---

**Alternate-byte phase**

In the alternate-bytes phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured in ABSIZE[1:0] of XSPI_CCR. The bytes to be sent are specified in XSPI_ABR.

The alternate-byte phase can send:
- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode, or in 16-bit SPI mode)

This can be configured using ABMODE[2:0] of XSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR of XSPI_CCR.

When ABMODE[2:0] = 000, the alternate-bytes phase is skipped and the command sequence proceeds directly to the next phase, if any.

Only a single nibble may need to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.
In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011), and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in XSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0], while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in XSPI_WABR, and the alternate byte format is specified in XSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in XSPI_ABR and XSPI_CCR.

**Dummy-cycle phase (memory latency)**

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when the higher clock frequencies are used. The number of cycles given during this phase is specified in DCYC[4:0] of XSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-, quad-, octal-, or 16-bit SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in XSPI_WTCR. The dummy cycles for the read operation are specified in XSPI_TCR.

**Data phase**

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in XSPI_DLR. In this mode, the data to be sent to the external device must be written to XSPI_DR. In indirect-read mode, the data received from the external device is obtained by reading XSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in XSPI_DLR, and the data received from the external device can be obtained by reading XSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AXI to the Cortex core or to a DMA.

The data phase can send/receive:
- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)
- 16 bits at a time (over IO0 to IO15 in 16-bit SPI mode)

This can be configured using DMODE[2:0] of XSPI_CCR.
The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR of XSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

In memory-mapped mode, the data format for the write operation is specified in XSPI_WCCR. The data format for the read operation is specified in XSPI_CCR.

**DQS use**

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE of XSPI_CCR.

**24.4.6 XSPI regular-command protocol signal interface**

**Single-SPI mode**

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, IO/SO and I1/SI are respectively equivalent to MOSI and MISO, having the XSPI generating the clock.

The different phases can each be configured separately to use this single-bit mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in single-SPI mode:
- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0 (to deactivate the “write protect” function).
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO15 are in output mode and forced to 0.
This is the case even for the dummy phase if DMODE[2:0] = 001.

**Dual-SPI mode**

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals. The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in dual-SPI mode:
- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO15 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 010, IO0/IO1 are always high-impedance.

**Quad-SPI mode**

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals. The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in quad-SPI mode:
- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO15 are in output mode and forced to 0.

In the dummy phase when DMODE[2:0] = 011, IO0 to IO3 are all high-impedance.

**Octal-SPI mode**

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals. The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 100, IO[0:7] are all high-impedance.

**XSPI mode**

In XSPI mode, the 16 bits are sent/received simultaneously over the IO[0:15] signals during the data phase.
The following phases must be configured separately to use the XSPI mode:

1. Set to 100 the IMODE, ADMODE, and ABMODE fields (in XSPI_CCR and XSPI_WCCR).
2. Set to 101 the DMODE fields (in XSPI_CCR and XSPI_WCCR).

In each phase that is configured in XSPI mode, IO[0:15] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase when DMODE[2:0] = 101, IO[0:15] are all high-impedance.

IO[8:15] are used only in XSPI mode. If none of the phases are configured to use this mode, then the pins corresponding to IO[8:15] can be used for other functions, even while the XSPI is active.

**Single-data rate (SDR) mode**

By default, all the phases operate in single-data rate (SDR) mode.

In SDR mode, when the XSPI drives the IO0/SO and IO1 to IO15 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the XSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in XSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

**Double-transfer rate (DTR) mode**

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in XSPI_CCR.

In memory-mapped mode, the DTR mode for each phase of the write operations is specified in XSPI_WCCR. The DTR mode for each phase of the read operations is specified in XSPI_CCR.

In DTR mode, when the XSPI drives the IO0/SO and IO1 to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

In DTR mode, when the XSPI drives the IO0 to IO15 signals in the data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the XSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in XSPI_CCR, the software must clear SSHIFT in XSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).
Dual-quad configuration

When \( \text{DMM} = 1 \) in \( \text{XSPI\_CR} \), the XSPI is in dual-memory configuration: if \( \text{DMODE} = 011 \), two external quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either SDR or DTR mode.

The device size, as specified in \( \text{DEVSIZE}[4:0] \) of \( \text{XSPI\_DCR1} \), must reflect the total external device capacity that is the double of the size of one individual component.

If address \( X \) is even, then the byte that the XSPI gives for address \( X \) is the byte at the address \( X/2 \) of device A, and the byte that the XSPI gives for address \( X + 1 \) is the byte at the address \( X/2 \) of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 2 bytes (16 bits), and the XSPI receives one byte from each device.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The forth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of \( \text{DL}[31:0] \) in \( \text{XSPI\_DLR} \) is stuck at 1 when \( \text{DMM} = 1 \).

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AI0x and the BI0x buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.
Dual-octal configuration

When DMM = 1 in XSPI_CCR, the XSPI is in dual-memory configuration: when DMODE = 100, two external octal-SPI devices (device A and device B) are used in order to receive 32 bits in DTR mode every cycle, effectively doubling the throughput as well as the capacity.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO7 signals.

The dual-octal configuration can be used in DTR mode exclusively in conjunction with the single-SPI, dual-SPI, quad-SPI and octal-SPI modes.

The device size, as specified in DEVSIZE[4:0] of XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-octal DTR mode, twice as many bytes must be read compared to the same read in regular DTR mode: if each device gives twice eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 4 bytes. The LSB is the LSB of device A, and the third byte is the LSB of device B.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-octal DTR mode. In such case, the order of retrieved status bits is as follows:
- first byte: LSB of device A
- second byte: second byte of device A
- third byte: LSB of device B
- fourth byte: second byte of device B

In indirect mode using DTR mode, a number of bytes multiple of four must always be accessed in XSPI mode. For this reason, bit 0 and bit 1 of the DL[31:0] bitfield in XSPI_DLR are stuck at 0 when DMODE[2:0] = 101.

In dual-octal configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

Note: The variable latency is not supported in dual-octal configuration.

XSPI mode

When DMODE[2:0] = 0b101 in XSPI_CCR, the XSPI is in single 16-bit-memory configuration: when DMODE[2:0] = 0b101 with DMM value ignored in that case. A single external XSPI device is used in order to send/receive 16 bits (or 32 bits in DTR mode) every cycle, effectively doubling the throughput.

The device provides/receives two separate DQS signals: DQS0 for the eight LSBs and DQS1 for the eight MSBs.
The XSPI mode can be used in conjunction with the single-SPI, dual-SPI, quad-SPI and octal-SPI modes, as well as with either the SDR or the DTR mode.

The device size, as specified in DEVSIZE[4:0] of XSPI_DCR1, must reflect the total external device capacity.

In SDR mode, a number of bytes multiple of two must always be accessed in XSPI mode. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 0 when DMODE = 101.

In DTR mode, a number of bytes multiple of four must always be accessed in XSPI mode. For this reason, bit 0 and 1 of DL[31:0] in XSPI_DLR are stuck at 0 when DMODE = 101.

### 24.4.7 HyperBus protocol

The XSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins in 8-bit data mode, or 19 to 20 pins in 16-bit data mode depending on the operating voltage:

- IO[7:0] as bidirectional data bus for 8-bit data mode and IO[15:0] as bidirectional data bus for 16-bit data mode
- DQS for read and write data strobe and latency insertion
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- command/address phase
- data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

---

**Figure 200. Example of HyperBus read operation (8-bit data mode)**

- NCS
- CLK
- DQS0
- IO[7:0]
- Dout A
- Dout B
- Dout[11:0]
- Dout[11:0]
- Memory drives IO[7:0] and DQS0
- Host drives IO[7:0] and memory drives DQS0
- Command-Address
- Latency count
- High = 2x Latency count
- Low = 1x Latency count

Note: DQS0 and data are edge aligned.
The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

**Command/address phase**

During this initial phase, the XSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

<table>
<thead>
<tr>
<th>CA bit</th>
<th>Bit name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>R/W#</td>
<td>Identifies the transaction as a read or a write</td>
</tr>
<tr>
<td>46</td>
<td>Address space</td>
<td>Indicates if the transaction accesses the memory or the register space</td>
</tr>
<tr>
<td>45</td>
<td>Burst type</td>
<td>Indicates if the burst is linear or wrapped</td>
</tr>
<tr>
<td>44-16</td>
<td>Row and upper column address</td>
<td>Selects the row and the upper column addresses</td>
</tr>
<tr>
<td>15-3</td>
<td>Reserved</td>
<td>-</td>
</tr>
<tr>
<td>2-0</td>
<td>Lower column address</td>
<td>Selects the starting 16-bit word within the half page</td>
</tr>
</tbody>
</table>

The address space is configured through the memory type MTYP[2:0] of XSPI_DCR1. The total size of the device must be considered through DEVSIZE[4:0] of XSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

---

**Warning:** Some memory specifications consider that each address corresponds to a 16-bit value. The XSPI considers that each address corresponds to an 8-bit value. So, the software needs to multiple the address by two when accessing the memory registers.

---

**Read/write operation with initial latency**

The HyperBus read and write operations need to respect two timings:

- **tRWR**: minimal read/write recovery time for the device (defined by TRWR[7:0] in XSPI_HLCR)
- **tACC**: access time for the device (defined in TACC[7:0] of XSPI_HLCR) according to the memory latency

During the read operation, the DQS0/1 is used by the device, in two ways (see Figure 200):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing
During the write operation, the DQS0/1 is used:
- by the device, during the command/address phase, to request an additional latency.
- by the XSPI, during the data phase, for write data masking.

**Figure 201. HyperBus write operation with initial latency (8-bit data mode)**

Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS0/1 must be tied to one during one of the DQS signals, during the command/address phase.

An additional $t_{\text{ACC}}$ duration is added by the XSPI to meet the device request.

**Figure 202. HyperBus read operation with additional latency (8-bit data mode)**
Fixed latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

The latency can be forced to 2 x t_{ACC} by setting LM of XSPI_HLCR.

In this XSPI latency mode, the state of the DQS signal is not taken into account by the XSPI and an additional latency is always added leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in XSPI_HLCR.
Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the \( t_{ACC} \) time.

**Figure 205. HyperBus read operation page crossing with latency (8-bit data mode)**

16-bit data transfer using HyperBus

In HyperBus protocol, the XSPI supports a dual-octal configuration (16-bit data transfers) when \( DMM = 1 \) in XSPI\_CR: one octal HyperBus memory is connected to IO0-IO7 and another is connected to IO8-IO15. These memories share all signals except DQS that are dedicated.

For 16-bit data transfers, \( DMODE[2:0] \) must be to equal to 101. Any other value in \( DMODE[2:0] \) correspond to 8-bit data transfer (quad-, dual-, and single-bit data transfer do not exist in HyperBus protocol). Command-address phase is always using 8 bits in HyperBus protocol (from IO8-IO15). Only the data is on 16-bit for write or read operations accessing the memory space (from IO0-IO15) as shown for instance in **Figure 206**. For the memory register accesses, the data is on 8-bit (from IO0 to IO7, IO8 to IO15 being not used but driven by the controller) as shown in **Figure 204**.

The behavior of the interface at protocol-level is exactly the same as for HyperBus octal configuration, as described above, except that the variable latency is not supported in dual-octal HyperBus configuration. LM in XSPI\_HLCR must be set.
24.4.8 Specific features

The XSPI supports some specific features, such as:
- wrap support
- NCS boundary and refresh
- communication regulation

Wrap support

The XSPI supports an hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in XSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 8 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated registers XSPI_WPxxx must be programmed according to the wrapped-read access characteristics. The dedicated XSPI_WPxxx registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (XSPI_CCR, XSPI_TCR, XSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.
NCS boundary and refresh

Two processes can be activated to regulate the XSPI transactions:

- NCS boundary
- refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with is configured in CSBOUND[4:0] of XSPI_DCR3 and it is equal to $2^{\text{CSBOUND}}$.

As an example, if CSBOUND[4:0] = 0x4, the boundary is set to $2^4 = 16$ bytes. As a consequence, the NCS is released each time that the LSB address is equal to 0xF, and each time that a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed in REFRESH[31:0] of XSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command, address, alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

Communication regulation

The communication regulation feature is useful in case two or more XSPIs share access to the same I/O port through an I/O manager. This feature limits the maximum length of a transaction to the value programmed in MAXTRAN[7:0] of XSPI_DCR3.

If the number of clock cycles of XSPI1 for example reach the MAXTRAN + 1 value, and if the second interface connected to the I/O manager requests an access, the NCS is released, and a new transaction is issued. If the second interface connected to the I/O manager does not request an access, the transaction is not stopped, and the NCS is not released.

If MAXTRAN[7:0] = 0, no limitation occurs.

The MAXTRAN[7:0] value must be greater than the minimal transaction size in terms of number of cycles including the command, address, alternate, and dummy phases.

Note: The communication regulation feature cannot be used in write mode for the flash memory devices that require extra command to reenable the write operation after the NCS is active again.

If NCS boundary, refresh and communication regulation are enabled at the same time, the NCS is released on the first condition met.

Restarting after an interrupted transfer

When a read or write operation is interrupted by a timeout, the XSPI interface, as soon as possible after getting back the port ownership, re-issues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.
24.4.9  XSPI operating modes introduction

The XSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):
- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

24.4.10  XSPI indirect mode

In indirect mode, the commands are started by writing to the XSPI registers, and the data is transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in XSPI_CR, the XSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to XSPI_DR.

When FMODE[1:0] = 01, the XSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading XSPI_DR.

In indirect mode, when the XSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The XSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in XSPI_DLR:
- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the XSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in XSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the XSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, TCF bit is set in XSPI_SR and an interrupt is generated if TCIE = 1 in XSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in XSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the XSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:
- if no address is necessary (ADMODE[2:0] = 000) and if no data needs to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in XSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data needs to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in XSPI_AR
• if data needs to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in XSPI_DR

A write to XSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in XSPI_SR.

**Triggering the start of a transfer in HyperBus protocol**

Depending on the XSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (XSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01)
- when a write is performed to DATA[31:0] (XSPI_DR) in indirect-write mode (when FMODE = 00)
- when a (dummy) write is performed to INSTRUCTION[31:0] (XSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01)

As soon as a transfer is started, the BUSY bit (XSPI_SR[5]) is automatically set.

**FIFO and data management**

Data in indirect mode pass through a 64-byte FIFO that is internal to the XSPI. FLEVEL in XSPI_SR indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in the XSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to the XSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

In indirect-read mode, when the FIFO becomes full, the XSPI temporarily stops reading bytes from the external device to avoid an overrun. Note that the reading of the external device does not restart until 4 bytes become vacant in the FIFO (when FLEVEL ≤ (FIFO_size - 4)). Thus, when FTHRES ≥ (FIFO_size - 3), the application must be sure to read enough bytes to assure that the XSPI starts retrieving data from the external device again. Otherwise, the FTF flag stays at 0 as long as FLEVEL < FTHRES.
The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

24.4.11 XSPI automatic status-polling mode

In automatic status-polling mode, the XSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in XSPI_SR goes high at this point, and stays high even between the periodic accesses.

The content of MASK[31:0] in XSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:
- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in XSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in XSPI_CR, the AND-match mode is activated: SMF is set in XSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in XSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in XSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the XSPI is disabled (EN = 0).

XSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in XSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. As a consequence, the memory must be configured in fixed latency.

24.4.12 XSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

**Note:** No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256-Mbyte range, then an AXI error is given. The effect of this error depends on the AXI master that attempted the access:
- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.
A support for execute in place (XIP) operation is implemented, where the XSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the XSPI never stops its prefetch operation, it either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time.

Since external devices tend to consume more when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in XSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in XSPI_LPTR, when x cycles have elapsed without an access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

### 24.4.13 XSPI configuration introduction

The XSPI configuration is done in three steps:

1. XSPI system configuration
2. XSPI device configuration
3. XSPI mode configuration

### 24.4.14 XSPI system configuration

The XSPI is configured using XSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- DMA use with DMAEN
- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in XSPI_LPTR.

The DMA channel must not be enabled during the XSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and XSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

### 24.4.15 XSPI device configuration

The parameters related to the external device targeted are configured through XSPI_DCR1 and XSPI_DCR2. The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

Number of bytes in the device = \(2^{(DEVSIZE+1)}\)

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the XSPI executes two commands, one immediately after the other, it raises the NCS high between the two commands, at least one CLK cycle by default.

If the external device requires more time between commands, CSHT[5:0] can be used to specify the minimum number of CLK cycles (up to 64) for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (tACC and tRWR) and the Latency mode must be configured in XSPI_HLCR.

**Memory types**

External memory providers may present some architecture and slight data formatting differences between them. The bitfield MTYP[2:0] into the XSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. This is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in a such way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the
memory. Refer to Figure 197: DTR read in octal-SPI mode with DQS (Macronix mode) example.

- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to Table 212: OctaRAM command address bit assignment (based on 64 Mb OctaRAM)). This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock when reading data from the memory.

Figure 208. OctaRAM read operation with reverse data ordering D1/D0

Table 212. OctaRAM command address bit assignment (based on 64 Mb(1) OctaRAM)

<table>
<thead>
<tr>
<th>Clock</th>
<th>1st clock</th>
<th>2nd clock</th>
<th>3rd clock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Command</td>
<td>Row address</td>
<td>Column address</td>
</tr>
<tr>
<td>SIO[7]</td>
<td>Reserved</td>
<td>RA7</td>
<td>CA9</td>
</tr>
<tr>
<td>SIO[6]</td>
<td>Reserved</td>
<td>RA6</td>
<td>CA8</td>
</tr>
<tr>
<td>SIO[5]</td>
<td>Reserved</td>
<td>RA5</td>
<td>CA7</td>
</tr>
<tr>
<td>SIO[4]</td>
<td>RA12</td>
<td>RA4</td>
<td>CA6</td>
</tr>
<tr>
<td>SIO[3]</td>
<td>RA11</td>
<td>RA3</td>
<td>CA5</td>
</tr>
<tr>
<td>SIO[2]</td>
<td>RA10</td>
<td>RA2</td>
<td>CA4</td>
</tr>
<tr>
<td>SIO[1]</td>
<td>RA9</td>
<td>RA1</td>
<td>Reserved</td>
</tr>
<tr>
<td>SIO[0]</td>
<td>RA8</td>
<td>RA0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

1. Example of 64 Mb OctaRAM address assignment:
   Row Address [RA12:RA0]: 8K. Column address [CA9:CA0]: 1K. 64 Mb density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes
must be used. The configuration to be set for this particular register space access
is MTYP[2:0] = 101.

- The software must configure MTYP[2:0] = 110 when the memory targeted comes from
APmemory and DMODE[2:0] = 101 to fit with the memory provider requirements
concerning the address formatting.
- Standard mode. It is the mode to use whenever the targeted memory is not
  corresponding to any others configurations described in this section. MTYP[2:0] = 010
for this standard mode.

24.4.16 XSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the
external device. When FMODE[1:0] = 01, the indirect-read mode is selected and data can
be read from the external device.

When the XSPI is used in indirect mode, the frames are constructed in the following way:
1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the frame timing in XSPI_TCR.
3. Specify the frame format in XSPI_CCR.
4. Specify the instruction in XSPI_IR.
5. Specify the optional alternate byte to be sent right after the address phase in
   XSPI_ABR.
6. Specify the targeted address in XSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

If neither the address register (XSPI_AR) nor the data register (XSPI_DR) need to be
updated for a particular command, then the command sequence starts as soon as XSPI_IR
is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just
ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to
be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as
soon as the address is updated with a write to XSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the
communication start is triggered by a write in the FIFO through XSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the
programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in
XSPI_DLR, it is ignored, and only 4 bytes are read. The periodicity is specified in XSPI_PIR.

Once the status data has been retrieved, the following can be processed:
- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.
The received value can be masked with the value stored in XSPI_PSMKR, and can be ORed or ANDed with the value stored in XSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The XSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in XSPI_DR.

When the XSPI is used in automatic status-polling mode, the frames are constructed in the following way:
1. Specify the input mask in XSPI_PSMKR.
2. Specify the comparison value in XSPI_PSMAR.
3. Specify the read period in XSPI_PIR.
4. Specify a number of data bytes to read in XSPI_DLR.
5. Specify the frame timing in XSPI_TCR.
6. Specify the frame format in XSPI_CCR.
7. Specify the instruction in XSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
9. Specify the optional targeted address in XSPI_AR.

If the address register (XSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as XSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

Memory-mapped mode is entered by setting FMODE[1:0] = 11 in XSPI_CR.

The programmed instruction and frame are sent when an AXI master accesses the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

When the XSPI is used in memory-mapped mode, the frames are constructed in the following way:
1. Specify the frame timing in XSPI_TCR for read operation.
2. Specify the frame format in XSPI_CCR for read operation.
3. Specify the instruction in XSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR for read operation.
5. Specify the frame timing in XSPI_WTCR for write operation.
6. Specify the frame format in XSPI_WCCR for write operation.
7. Specify the instruction in XSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_WABR for write operation.

All the configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 have no effect and is not signaled with an error response. On the first access, the XSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

Memory-mapped write operations are not supported without write strobe.

**XSPI delayed data sampling when no DQS is used**

By default, when no DQS is used, the XSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in XSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

**XSPI delayed data sampling when DQS is used**

When external DQS is used as a sampling clock, it is shifted precisely by one quarter of the SPI clock cycle, for all frequencies above freq_min, to compensate the data propagation delay in the “high-speed interface” when the product embeds one.

### 24.4.17 XSPI HyperBus protocol configuration

**Indirect mode configuration (HyperBus)**

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the XSPI is used in indirect mode, the frames are constructed in the following way:
1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the targeted address in XSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through XSPI_DR.
Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AXI master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the XSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

24.4.18 XSPI error management

The following errors set the TEF flag in XSPI_SR and generates an interrupt if enabled (TEIE = 1 in XSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in XSPI_AR (according to the device size defined by DEVSIZE[4:0]).
- in indirect mode, if the address plus the data length exceed the device size. TEF is set as soon as the access is triggered.

In memory-mapped mode, the XSPI generates an AXI slave error in the following situations:

- The memory-mapped mode is disabled and an AXI read or write request occurs.
- Read or write address exceeds the size of the external memory.
- Abort is received while a read or write burst is ongoing when the abort condition is present at the moment of transfer phases preceding the data phase (In incremental or wrap burst type modes). To avoid the error, refer to Section 24.4.20: XSPI BUSY and ABORT.
- The XSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in XSPI_WCCR in octal DTR mode, in dual-memory configuration, in hyperbus mode or 16-bit mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means AXI Bus transfer size (ARSIZE) is different from 8 bytes (only for memory-mapped mode).
- Bit DMM is set in XSPI_CR (dual-memory configuration) and 16-bit mode is set.

24.4.19 XSPI high-speed interface and calibration

To reach higher frequencies, a dedicated high-speed interface is inserted between the XSPI (or the I/O manager in case the product embeds one), and the I/O pads.
The following is valid for all data bus sizes 1, 2, 4, 8 or 16 bits.

The high-speed interface block embeds resynchronization registers that are clocked by delayed clock created from a DLL (delay locked loop) also located in the high-speed interface. The high-speed interface features are controlled by registers located in the XSPI.

The purpose of resynchronization is primary to shift data or data strobe by one quarter of octal bus clock period, with a correct timing accuracy. DLL must be calibrated versus this clock period.

The calibration process is automatically enabled when one of the three conditions below is met:

- The XSPI exits reset state.
- A value is written in PRESCALER[7:0] of XSPI_DCR2.
- A value is written in XSPI_CCR.

The calibration process starts when the two following conditions are both met:

- The calibration has been enabled by one of the three conditions above.
- An action that sets BUSY = 1 is performed. For example the first transfer to memory after calibration is enabled. When the calibration is completed, BUSY returns to 0.

In case a periodic recalibration is needed (for example to take in account possible variations in temperature or power supply on a long duration), this recalibration must be triggered by writing periodically in PRESCALER[7:0] of XSPI_DCR2, while BUSY = 0.

Once the calibration is completed, the value of the SPI bus clock period, expressed in number of unitary delay, is available to user in COARSE[5:0] and FINE[6:0] of XSPI_CALFCR.

After auto-calibration, XSPI_CALSOR and XSPI_CALSIR are automatically loaded with the same value that corresponds to the delay for a quarter cycle.

The automatic calibration is not executed if at least one of both registers XSPI_CALSOR/XSPI_CALSIR is written between the last write operation on XSPI_DCR2 or XSPI_CCR and the next transfer start. In such case, the calibration values must be set by the software code into XSPI_CALSOR and XSPI_CALSIR registers. It may improve the flash programming time without relaunching the automatic calibration processing between two programming pages.

When the memory is not supporting DQS (DQSE = 0), the automatic calibration is not used in reception. The DLL Master is used instead for delaying the feedback clock (XSPI_CALMR). This delay needs to be adjusted by the application itself, using a software sequence that determines which delay is optimal to guarantee the correct read operations.

When the clock is divided in DTR transmission mode, the quarter cycle delays on DQS/data are not inserted by the DLLs themselves, but by internal flops design scheme. In SDR transmission mode, the DLLs are not used and this, whatever the clock prescaler value.

In case of DTR mode and prescaler bypassed (PRESCALER[7:0] = 0), the kernel clock provided to interface must have a 50 % duty-cycle.

When using the high-speed-interface, the system clock (AXI clock) must be at least as fast as the SPI clock.
24.4.20 XSPI BUSY and ABORT

Once the XSPI starts an operation with the external device, BUSY is automatically set in XSPI_SR.

In indirect mode, BUSY is reset once the XSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in XSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers, otherwise, an error may be triggered and the data can be discarded. When DMA is enabled to handle the data read or write operations in XSPI_DR, it is recommended to disable the DMA channel before aborting the XSPI.

*Note:* Some devices may misbehave if a write operation to a status register is aborted.

24.4.21 XSPI reconfiguration or deactivation

Before any XSPI reconfiguration, the software must ensure that all the transactions are completed:

- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier then an abort.

24.4.22 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

*Figure 209. NCS when CKMODE = 0 (T = CLK period)*
When CKMODE = 1 (clock mode 3: CLK goes high when no operation is in progress) and when in SDR mode, NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

**Figure 210. NCS when CKMODE = 1 in SDR mode (T = CLK period)**

When the CKMODE = 1 (clock mode 3) and DDTR = 1 (data DTR mode), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final active rising CLK edge (see the figure below). Because the DTR operations must finish with a falling edge, the CLK is low when NCS rises, and CLK rises back up one half of a CLK cycle afterwards.

**Figure 211. NCS when CKMODE = 1 in DTR mode (T = CLK period)**

When the FIFO stays full during a read operation, or if the FIFO stays empty during a write operation, the operation stalls and CLK stays low until the software services the FIFO. If an abort occurs when an operation is stalled, NCS rises just after the abort is requested and then CLK rises one half of a CLK cycle later (see the figure below).

**Figure 212. NCS when CKMODE =1 with an abort (T = CLK period)**

In dual-memory configuration, the NCS2 behaves exactly the same as the NCS1. The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.
24.4.23 Software-controlled dual-memory support

One XSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the XSPI.

The XSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in XSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus. From a memory-map point of view, these two memories are located in the same address range.

![Example of software-controller dual-memory support](image)

24.5 Address alignment and data number

The following table summarizes the effect of the address alignment and programmed data number depending on the use case.

<table>
<thead>
<tr>
<th>Memory type</th>
<th>Transaction type</th>
<th>Constraint on address(1)</th>
<th>Impact if constraint on address not respected</th>
<th>Constraint on number of bytes(1)</th>
<th>Impact if constraint on bytes not respected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single, dual, quad flash or SRAM (DMM = 0)</td>
<td>IND(2) read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM(3) read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Single, dual, quad flash or SRAM (DMM = 1)</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.</td>
<td>Even</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>Even</td>
<td>ADDR[0] is set to 0.</td>
<td>Even</td>
<td>DLR[0] is set to 1.</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>Even</td>
<td>Slave error</td>
<td>Even</td>
<td>Last byte is lost.</td>
</tr>
</tbody>
</table>
### Table 213. Address alignment cases (continued)

<table>
<thead>
<tr>
<th>Memory type</th>
<th>Transaction type</th>
<th>Constraint on address$^{(1)}$</th>
<th>Impact if constraint on address not respected</th>
<th>Constraint on number of bytes$^{(1)}$</th>
<th>Impact if constraint on bytes not respected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octal flash in SDR mode</td>
<td>IND read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octal memory in DTR mode without WDM$^{(6)}$, or 16-bit memory in SDR mode without WDM</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.$^{(4)}$</td>
<td>Even</td>
<td>DLR[0] is set to 1.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>Even</td>
<td>ADDR[0] is set to 0.$^{(4)}$</td>
<td>Even</td>
<td>DLR[0] is set to 1.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>16-bit or dual-octal memory in DTR mode without WDM</td>
<td>IND read</td>
<td>Aligned</td>
<td>ADDR[1:0] is assumed to be 00.$^{(4)}$</td>
<td>N × 4</td>
<td>DLR[1:0] is assumed to be 11.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write</td>
<td>Aligned</td>
<td>ADDR[1:0] is assumed to be 00.$^{(4)}$</td>
<td>N × 4</td>
<td>DLR[1:0] is assumed to be 11.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Octal flash or RAM in DTR mode with WDM</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.$^{(4)}$</td>
<td>Even</td>
<td>DLR[0] is set to 1.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write$^{(7)}$</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HyperBus</td>
<td>IND read</td>
<td>Even</td>
<td>ADDR[0] is set to 0.$^{(4)}$</td>
<td>Even</td>
<td>DLR[0] is set to 1.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write$^{(7)}$</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit data bus memory or dual-octal memory with WDM</td>
<td>IND read</td>
<td>Aligned</td>
<td>ADDR[1:0] is assumed to be 00.$^{(4)}$</td>
<td>N × 4</td>
<td>DLR[1:0] is assumed to be 11.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>IND write$^{(8)}$</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-bit HyperBus</td>
<td>IND read</td>
<td>Aligned</td>
<td>ADDR[1:0] is assumed to be 00.$^{(4)}$</td>
<td>N × 4</td>
<td>DLR[1:0] is assumed to be 11.$^{(5)}$</td>
</tr>
<tr>
<td></td>
<td>MM read</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IND write$^{(9)}$</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>MM write</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode.
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.

7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

8. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 4 aligned in DTR mode or modulo 2 in SDR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

9. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 4 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

### 24.6 XSPI interrupts

An interrupt can be produced on the following events:
- timeout
- status match
- FIFO threshold
- transfer complete
- transfer error

Separate interrupt enable bits are available to provide more flexibility.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>TOF</td>
<td>TOIE</td>
</tr>
<tr>
<td>Status match</td>
<td>SMF</td>
<td>SMIE</td>
</tr>
<tr>
<td>FIFO threshold</td>
<td>FTF</td>
<td>FTIE</td>
</tr>
<tr>
<td>Transfer complete</td>
<td>TCF</td>
<td>TCIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TEF</td>
<td>TEIE</td>
</tr>
</tbody>
</table>

### Table 214. XSPI interrupt requests

<table>
<thead>
<tr>
<th>MSEL[1:0]</th>
<th>FMODE[1:0]</th>
<th>CSSEL</th>
<th>PMM</th>
<th>APMS</th>
<th>TOIE</th>
<th>SMIE</th>
<th>FTHRES[5:0]</th>
<th>DMM</th>
<th>TOIE</th>
<th>DMAEN</th>
<th>ABORT</th>
<th>EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

### 24.7 XSPI registers

#### 24.7.1 XSPI control register (XSPI_CR)

Address offset: 0x0000
Reset value: 0x00000 0000
Bits 31:30 **MSEL[1:0]: Flash select**  
These bits select the memory to be addressed in single-, dual-, quad-, or octal-SPI mode in single-memory configuration (when DMM = 0).  
- when in quad-SPI mode:  
  00: Data exchanged over IO[3:0]  
  01: Data exchanged over IO[7:4]  
  10: Data exchanged over IO[11:8]  
  11: Data exchanged over IO[15:12]  
- when in octal-SPI mode or dual-quad configuration:  
  0x: Data exchanged over IO[7:0]  
  1x: Data exchanged over IO[15:8]  
These bits are ignored when in dual-octal configuration (data on 8 bits and DMM = 1) or 16-bit configuration (data exchanged over IO[15:0]).  
*Note: Bit 30 is mirrored in bit 7. This bitfield can be modified only when BUSY = 0.*

Bits 29:28 **FMODE[1:0]: Functional mode**  
This bitfield defines the XSPI functional mode of operation.  
00: Indirect-write mode  
01: Indirect-read mode  
10: Automatic status-polling mode (relevant in regular-command protocol only)  
11: Memory-mapped mode  
If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.  
*Note: This bitfield can be modified only when BUSY = 0.*

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **CSSEL: chip select selection**  
This bit indicates if the XSPI must activate NCS1 or NCS2.  
0: NCS1 active  
1: NCS2 active  
*Note: This bit can be modified only when BUSY = 0.*

Bit 23 **PMM: Polling match mode**  
This bit indicates which method must be used to determine a match during the automatic status-polling mode.  
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.  
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.  
*Note: This bit can be modified only when BUSY = 0.*

Bit 22 **APMS: Automatic status-polling mode stop**  
This bit determines if the automatic status-polling mode is stopped after a match.  
0: Automatic status-polling mode is stopped only by abort or by disabling the XSPI.  
1: Automatic status-polling mode stops as soon as there is a match.  
*Note: This bit can be modified only when BUSY = 0.*

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE: Timeout interrupt enable**  
This bit enables the timeout interrupt.  
0: Interrupt disabled  
1: Interrupt enabled
Bit 19  **SMIE**: Status match interrupt enable
This bit enables the status match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18  **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17  **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16  **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

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

Bits 13:8  **FTHRES[5:0]**: FIFO threshold level
This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in XSPI_SR, to be set.
000000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.
000001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.
...
111111: FTF is set if there are 64 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 64 valid bytes can be read from the FIFO in indirect-read mode.

*Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[5:0] value.*

Bit 7  Reserved, must be kept at reset value.

Bit 6  **DMM**: Dual-memory configuration
This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity.
0: Dual-memory configuration disabled
1: Dual-memory configuration enabled

*Note: This bit can be modified only when BUSY = 0.*

Bits 5:4  Reserved, must be kept at reset value.

Bit 3  **TCEN**: Timeout counter enable
This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.
0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.
1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

*Note: This bit can be modified only when BUSY = 0.*
24.7.2 XSPI device configuration register 1 (XSPI_DCR1)

Address offset: 0x008
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:27 Reserved, must be kept at reset value.
Bits 26:24 **MTYP[2:0]:** Memory type

This bitfield indicates the type of memory to be supported.

- 000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes. In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.
- 010: Standard mode
- 011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).
- 100: HyperBus memory mode, the protocol follows the HyperBus specification.
- 101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect read/write modes must be used.
- 110: AP memory mode. If DMODE = 101, there is a special hardware operation on address word from the bit 10 and above to fit the provider requirement (shift operation on the left from the address bit 10, keeping this last at 0).
- Others: Reserved

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **DEVSIZE[4:0]:** Device size

This bitfield defines the size of the external device using the following formula:

Number of bytes in device = 2^{DEVSIZE+1}.

DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

In regular-command protocol, if DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

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

Bits 13:8 **CSHT[5:0]:** Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.

- 0x0: NCS stays high for at least 1 cycle between external device commands.
- 0x1: NCS stays high for at least 2 cycles between external device commands.
- ...
- 0x3F: NCS stays high for at least 64 cycles between external device commands.

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

**Bit 1 **FRCK:** Free running clock

This bit configures the free running clock.

- 0: CLK is not free running.
- 1: CLK is free running (always provided).

**Bit 0 **CKMODE:** Clock mode 0/mode 3

This bit indicates the level taken by the CLK between commands (when NCS = 1).

- 0: CLK must stay low while NCS is high (chip-select released), referred to as clock mode 0.
- 1: CLK must stay high while NCS is high (chip-select released), referred to as clock mode 3.
24.7.3  XSPI device configuration register 2 (XSPI_DCR2)

Address offset: 0x00C
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:19  Reserved, must be kept at reset value.

Bits 18:16  WRAPSIZE[2:0]: Wrap size
This bitfield indicates the wrap size to which the memory is configured. For memories, which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in XSPI_WPIR.
- 000: Wrapped reads are not supported by the memory.
- 010: External memory supports wrap size of 16 bytes.
- 011: External memory supports wrap size of 32 bytes.
- 100: External memory supports wrap size of 64 bytes.
- 101: External memory supports wrap size of 128 bytes.
- Others: Reserved

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  PRESCALER[7:0]: Clock prescaler
This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).
- 0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as XSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the XSPI a kernel clock that has 50% duty-cycle.
- 1: $F_{CLK} = F_{KERNEL}/2$
- 2: $F_{CLK} = F_{KERNEL}/3$
- ... 
- 255: $F_{CLK} = F_{KERNEL}/256$
For odd clock division factors, the CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.
Writing this bitfield automatically starts a new calibration of high-speed interface DLL at the start of next transfer, except in case XSPI_CALOSR or XSPI_CALISR have been written in the meantime. BUSY stays high during the whole calibration execution.
### 24.7.4 XSPI device configuration register 3 (XSPI_DCR3)

Address offset: 0x010  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary  
This bitfield enables the transaction boundary feature. When active, a minimum value of three is recommended.  
The NCS is released on each boundary of $2^{CSBOUND}$ bytes.  
0: NCS boundary disabled  
Others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **MAXTRAN[7:0]**: Maximum transfer  
This bitfield enables the communication regulation feature.  
The NCS is released every MAXTRAN+1 clock cycles when the other XSPI request the access to the bus.  
0: Maximum communication disabled  
Others: Maximum communication is set to (MAXTRAN + 1) bytes.

### 24.7.5 XSPI device configuration register 4 (XSPI_DCR4)

Address offset: 0x014  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

**REFRESH[31:16]**  
**REFRESH[15:0]**
24.7.6  XSPI status register (XSPI_SR)

Address offset: 0x020
Reset value: 0x0000 0000

Bits 31:0  REFRESH[31:0]: Refresh rate
This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual-, or quad-SPI mode, because the byte transmission must be completed.
0: Refresh disabled
Others: Maximum communication length is set to REFRESH + 1 clock cycles.

Bits 31:15  Reserved, must be kept at reset value.

Bits 14:8  FLEVEL[6:0]: FIFO level
This bitfield gives the number of valid bytes that are being held in the FIFO. FLEVEL = 0 when the FIFO is empty, and 64 when it is full. In automatic-status polling mode, FLEVEL is zero.

Bits 7:6  Reserved, must be kept at reset value.

Bit 5  BUSY: Busy
This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4  TOF: Timeout flag
This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3  SMF: Status match flag
This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (XSPI_PSMAR).
It is cleared by writing 1 to CSMF.

Bit 2  FTF: FIFO threshold flag
In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.
It is cleared automatically as soon as the threshold condition is no longer true.
In automatic status-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1  TCF: Transfer complete flag
This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted. It is cleared by writing 1 to CTCF.

Bit 0  TEF: Transfer error flag
This bit is set in indirect mode when an invalid address is being accessed in indirect mode.
It is cleared by writing 1 to CTEF.
### 24.7.7 XSPI flag clear register (XSPI_FCR)

Address offset: 0x024  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 31:5: Reserved, must be kept at reset value.
- Bit 4 **CTOF**: Clear timeout flag  
  Writing 1 clears the TOF flag in the XSPI_SR register.
- Bit 3 **CSMF**: Clear status match flag  
  Writing 1 clears the SMF flag in the XSPI_SR register.
- Bit 2: Reserved, must be kept at reset value.
- Bit 1 **CTCF**: Clear transfer complete flag  
  Writing 1 clears the TCF flag in the XSPI_SR register.
- Bit 0 **CTEF**: Clear transfer error flag  
  Writing 1 clears the TEF flag in the XSPI_SR register.

### 24.7.8 XSPI data length register (XSPI_DLR)

Address offset: 0x040  
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

*STI*
Bits 31:0  **DL[31:0]:** Data length
Number of data to be retrieved (value + 1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.
All 1’s in indirect mode means undefined length, where XSPI continues until the end of the memory, as defined by DEVSIZE.
0x0000_0000: 1 byte is to be transferred.
0x0000_0001: 2 bytes are to be transferred.
0x0000_0002: 3 bytes are to be transferred.
0x0000_0003: 4 bytes are to be transferred.
... 0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.
0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.
0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.
DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.
This bitfield has no effect when in memory-mapped mode.

24.7.9  **XSPI address register (XSPI_AR)**
Address offset: 0x048
Reset value: 0x0000 0000
This register can be modified only when BUSY = 0 and FMODE ≠ 11.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **ADDRESS[31:0]:** Address
Address to be sent to the external device. In HyperBus protocol, this bitfield must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 0.

**Caution:** Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiple the address by two when accessing the memory registers.
### 24.7.10 XSPI data register (XSPI_DR)

Address offset: 0x050

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>DATA[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:0** `DATA[31:0]`: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read `DATA[7:0]` and a half-word read must read `DATA[15:0]`.

### 24.7.11 XSPI polling status mask register (XSPI_PSMKR)

Address offset: 0x080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>MASK[31:16]</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:0** `MASK[31:0]`: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.
### 24.7.12 XSPI polling status match register (XSPI_PSMAR)

Address offset: 0x088  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Format</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>MATCH[31:16]: Status match</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>Value to be compared with the masked status register to get a match</td>
</tr>
<tr>
<td>15:0</td>
<td>MATCH[15:0]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

### 24.7.13 XSPI polling interval register (XSPI_PIR)

Address offset: 0x090  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Format</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>INTERVAL[31:16]: Reserved, must be kept at reset value.</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>15:0</td>
<td>INTERVAL[15:0]: Polling interval</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>Number of CLK cycles between a read during the automatic status-polling phases</td>
</tr>
</tbody>
</table>

### 24.7.14 XSPI communication configuration register (XSPI_CCR)

Address offset: 0x100  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.
Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
- 0: DQS disabled
- 1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
- 0: DTR mode disabled for the data phase
- 1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
- 000: No data
- 001: Data on a single line
- 010: Data on two lines
- 011: Data on four lines
- 100: Data on eight lines
- 101: Data on 16 lines
- Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
- 00: 8-bit alternate bytes
- 01: 16-bit alternate bytes
- 10: 24-bit alternate bytes
- 11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
- 0: DTR mode disabled for the alternate-byte phase
- 1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
- 000: No alternate bytes
- 001: Alternate bytes on a single line
- 010: Alternate bytes on two lines
- 011: Alternate bytes on four lines
- 100: Alternate bytes on eight lines
- Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
- 00: 8-bit address
- 01: 16-bit address
- 10: 24-bit address
- 11: 32-bit address
24.7.15 XSPI timing configuration register (XSPI_TCR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

Bit 11 **ADDTR**: Address double transfer rate

This bit sets the DTR mode for the address phase.

0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode

This bitfield defines the address phase mode of operation.

000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size

This bitfield defines the instruction size.

00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate

This bit sets the DTR mode for the instruction phase.

0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode

This bitfield defines the instruction phase mode of operation.

000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

Bit 31 Reserved, must be kept at reset value.
24.7.16 XSPI instruction register (XSPI_IR)

Address offset: 0x110
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 INSTRUCTION[31:0]: Instruction

Instruction to be sent to the external SPI device

24.7.17 XSPI alternate bytes register (XSPI_ABR)

Address offset: 0x120
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 ALTERNATE[31:0]: Alternate bytes

Optional data to be sent to the external SPI device right after the address.
24.7.18  XSPI low-power timeout register (XSPI_LPTR)

Address offset: 0x130
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TIMEOUT[15:0]**

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **TIMEOUT[15:0]:** Timeout period

After each access in memory-mapped mode, the XSPI prefetches the subsequent bytes and hold them in the FIFO.

This bitfield indicates how many CLK cycles the XSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

24.7.19  XSPI wrap communication configuration register (XSPI_WPCCR)

Address offset: 0x140
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

**DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase
Bits 26:24  **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
101: Data on 16 lines
Others: Reserved

Bits 23:22  Reserved, must be kept at reset value.

Bits 21:20  **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19  **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16  **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12  **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11  **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8  **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved
Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size

- This bitfield defines the instruction size.
- 00: 8-bit instruction
- 01: 16-bit instruction
- 10: 24-bit instruction
- 11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate

- This bit sets the DTR mode for the instruction phase.
- 0: DTR mode disabled for the instruction phase
- 1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode

- This bitfield defines the instruction phase mode of operation.
- 000: No instruction
- 001: Instruction on a single line
- 010: Instruction on two lines
- 011: Instruction on four lines
- 100: Instruction on eight lines
- Others: Reserved

### 24.7.20 XSPI wrap timing configuration register (XSPI_WPTCR)

Address offset: 0x148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSSHIFT**: Sample shift

- By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
- This bit allows the data to be sampled later in order to consider the external signal delays.
- 0: No shift
- 1: 1/2 cycle shift
- The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1).

Bits 29:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles

- This bitfield defines the duration of the dummy phase according to the memory latency.
- In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).
24.7.21 **XSPI wrap instruction register (XSPI_WPIR)**

Address offset: 0x150  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
<th>Offset (Address)</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>INSTRUCTION[31:16]</td>
<td>rw</td>
<td>0x150</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>15:0</td>
<td>INSTRUCTION[15:0]</td>
<td>rw</td>
<td>0x150</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

Bits 31:0 **INSTRUCTION[31:0]**: Instruction  
Instruction to be sent to the external SPI device

24.7.22 **XSPI wrap alternate byte register (XSPI_WPABR)**

Address offset: 0x160  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
<th>Offset (Address)</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>ALTERNATE[31:16]</td>
<td>rw</td>
<td>0x160</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>15:0</td>
<td>ALTERNATE[15:0]</td>
<td>rw</td>
<td>0x160</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes  
Optional data to be sent to the external SPI device right after the address

24.7.23 **XSPI write communication configuration register (XSPI_WCCR)**

Address offset: 0x180  
Reset value: 0x0000 0000  
This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Access</th>
<th>Offset (Address)</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RES RES DQSE DDTR DMODE[2:0]</td>
<td>rw</td>
<td>0x180</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td>15:0</td>
<td>RES RES ABSIZE[1:0] ABDTR ABMODE[2:0]</td>
<td>rw</td>
<td>0x180</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td></td>
<td>RES RES ADSIZE[1:0] ADDTR ADMODE[2:0]</td>
<td>rw</td>
<td>0x180</td>
<td>0x0000 0000</td>
</tr>
<tr>
<td></td>
<td>RES RES ISIZE[1:0] IDTR IMODE[2:0]</td>
<td>rw</td>
<td>0x180</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>
Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
- 0: DQS disabled
- 1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
- 0: DTR mode disabled for the data phase
- 1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
- 000: No data
- 001: Data on a single line
- 010: Data on two lines
- 011: Data on four lines
- 100: Data on eight lines
- 101: Data on 16 lines
- Others: Reserved

 Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size:
- 00: 8-bit alternate bytes
- 01: 16-bit alternate bytes
- 10: 24-bit alternate bytes
- 11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double-transfer rate
This bit sets the DTR mode for the alternate-byte phase.
- 0: DTR mode disabled for the alternate-byte phase
- 1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
- 000: No alternate bytes
- 001: Alternate bytes on a single line
- 010: Alternate bytes on two lines
- 011: Alternate bytes on four lines
- 100: Alternate bytes on eight lines
- Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
- 00: 8-bit address
- 01: 16-bit address
- 10: 24-bit address
- 11: 32-bit address
24.7.24 XSPI write timing configuration register (XSPI_WTCR)

Address offset: 0x188
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:5 Reserved, must be kept at reset value.
24.7.25  XSPI write instruction register (XSPI_WIR)

Address offset: 0x190
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
```

Bits 31:0  **INSTRUCTION[31:0]:** Instruction
Instruction to be sent to the external SPI device

24.7.26  XSPI write alternate byte register (XSPI_WABR)

Address offset: 0x1A0
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
```

Bits 31:0  **ALTERNATE[31:0]:** Alternate bytes
Optional data to be sent to the external SPI device right after the address
24.7.27  XSPI HyperBus latency configuration register (XSPI_HLCR)

Address offset: 0x200
Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  TRWR[7:0]: Read-write minimum recovery time
            Device Read-to-write/write-to-read minimum recovery time expressed in number of
            communication clock cycles

Bits 15:8  TACC[7:0]: Access time
            Device access time according to the memory latency, expressed in number of
            communication clock cycles

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

Bit 1  WZL: Write zero latency
            This bit enables zero latency on write operations.
            0: Latency on write accesses
            1: No latency on write accesses

Bit 0  LM: Latency mode
            This bit selects the latency mode.
            0: Variable initial latency
            1: Fixed latency
            Note:  This bit must be set when using the dual-octal HyperBus configuration.

24.7.28  XSPI full-cycle calibration configuration (XSPI_CALFCR)

Address offset: 0x210
Reset value: 0x0000 0000

This read-only register gives the calibration code needed by the DLL master so that its delay is
equivalent to a full memory-clock cycle. The value of this register is updated every time that auto-calibration finishes.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  COARSE[4:0]

Bits 15:8  FINE[6:0]
24.7.29 XSPI DLL master calibration configuration (XSPI_CALMR)

Address offset: 0x218
Reset value: 0x0000 0000

The DLL Master is used for delaying the feedback clock when reading without DQS.
The delay of the master DLL is determined by the value in this register.
This register can always be read by software and can be modified only when BUSY = 0.
This register never gets updated automatically by hardware.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>CALMAX: Max value</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit gets set when the memory-clock period is outside the range of DLL master, in which case XSPI_CALFCR and XSPI_CALSR are updated with the values for the maximum delay.</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 30:21 | Reserved, must be kept at reset value. |
| Bits 20:16 | COARSE[4:0]: Coarse calibration |
| The delay unitary value for this bitfield depends on product technology (see the datasheet). |

| Bits 15:7 | Reserved, must be kept at reset value. |
| Bits 6:0 | FINE[6:0]: Fine calibration |
| The delay unitary value for this bitfield depends on product technology (see the datasheet). |

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>CALMAX: Max value</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit gets set when the memory-clock period is outside the range of DLL master, in which case XSPI_CALFCR and XSPI_CALSR are updated with the values for the maximum delay.</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 30:21 | Reserved, must be kept at reset value. |
| Bits 20:16 | COARSE[4:0]: Coarse calibration |
| The delay unitary value for this bitfield depends on product technology (see the datasheet). |

| Bits 15:7 | Reserved, must be kept at reset value. |
| Bits 6:0 | FINE[6:0]: Fine calibration |
| The delay unitary value for this bitfield depends on product technology (see the datasheet). |
24.7.30 XSPI DLL slave output calibration configuration (XSPI_CALSIR)

Address offset: 0x220
Reset value: 0x0000 0000

The DLL output slave is used to delay the output data in DDR mode for write operations. The delay of the output slave DLL is determined by the value in this register.

This register is updated automatically by hardware at the end of calibration (at the same moment that XSPI_CALFCR is updated).

If this register is written after the last write to XSPI_DCR2 or XSPI_CCR, then auto-calibration is not executed on the next transfer (auto-calibration is not performed for XSPI_CALSIR as well).

This register can always be read by software and can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>COARSE[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw rw rw rw rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw rw rw rw rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>FINE[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **COARSE[4:0]**: Coarse calibration

The delay unitary value for this bitfield depends on product technology.

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

Bits 6:0 **FINE[6:0]**: Fine calibration

The delay unitary value for this bitfield depends on product technology.
24.7.31 XSPI DLL slave input calibration configuration (XSPI_CALSIR)

Address offset: 0x228
Reset value: 0x0000 0000

The DLL input slave is used to delay the DQS input for sampling the data when DQS is enabled for read operations. The delay of the input slave DLL is determined by the value in this register.

This register is updated automatically by hardware at the end of calibration (at the same moment that XSPI_CALFCR is updated).

If this register is written after the last write to XSPI_DCR2 or XSPI_CCR, then auto-calibration is not executed on the next transfer (auto-calibration is not performed for XSPI_CALSOR as well).

This register can always be read by software and can be modified only when BUSY = 0.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>XSPI_CR</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0004</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>XSPI_DCR1</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 COARSE[4:0]: Coarse calibration
The delay unitary value for this bitfield depends on product technology.

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

Bits 6:0 FINE[6:0]: Fine calibration
The delay unitary value for this bitfield depends on product technology.

24.7.32 XSPI register map

Table 215. XSPI register map and reset values

| Offset | Register name         | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | XSPI_CR               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0004 | Reserved              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | XSPI_DCR1             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

1058/3791 RM0477 Rev 6
## Table 215. XSPI register map and reset values

| Offset | Register name | Offset name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00C  | XSPI_DCR2     | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x010  | XSPI_DCR3     | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x014  | XSPI_DCR4     | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x018- | XSPI_SR       | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x020  | XSPI_FCR      | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x028- | XSPI_DLR      | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x040  | XSPI_AR       | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04C  | XSPI_DR       | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x054- | XSPI_PSMKR    | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x080  | XSPI_PSMAR    | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08C  | XSPI_PIR      | RES          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

**Note:** The table includes reset values for each register, with reset conditions specified for each bit position.
### Extended-SPI interface (XSPI) RM0477

#### Table 215. XSPI register map and reset values

| Offset | Register name  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x100  | XSPI_CCR      |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x104  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x108  | XSPI_TCR      |    |    |    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10C  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x110  | XSPI_IR       |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x114  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x118  | XSPI_ABR      |    |    |    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x11C  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x130  | XSPI_LPTR     |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x134  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x140  | XSPI_WPCCR    |    |    |    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x144  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x148  | XSPI_WPTCR    |    |    |    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14C  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x150  | XSPI_WPIR     |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x154  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x160  | XSPI_WPABR    |    |    |    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x164  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x180  | XSPI_WCCR     |    |    |    | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Reset value 00000000000000000000000000000000

---

Table 215. XSPI register map and reset values
Refer to Section 2.3 for the register boundary addresses.
25  

**XSPI I/O manager (XSPIM)**

### 25.1 Introduction

The XSPI I/O manager is a low-level interface, enabling an efficient XSPI pin assignment with a full I/O matrix (before alternate function map), and multiplex of single/dual/quad/octal/16-bit SPI interfaces over the same bus.

### 25.2 XSPIM main features

- Supports up to two single/dual/quad/octal/16-bit SPI interfaces
- Supports up to two ports for pin assignment
- Supports high-speed interfaces

### 25.3 XSPIM implementation

*Table 216* describes the XSPIM implementation.

<table>
<thead>
<tr>
<th>XSPI feature</th>
<th>Available on the devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supports up to two single/dual/quad interfaces</td>
<td>X</td>
</tr>
<tr>
<td>Fully I/O multiplexing capability</td>
<td>X</td>
</tr>
<tr>
<td>Supports time-multiplexed mode</td>
<td>X</td>
</tr>
<tr>
<td>Supports high-speed interface</td>
<td>X</td>
</tr>
<tr>
<td>Chip select selection if XSPI provides dual chip select</td>
<td>X</td>
</tr>
<tr>
<td>Supports 16-bit data interface and dual-octal mode</td>
<td>X</td>
</tr>
</tbody>
</table>

### 25.4 XSPIM functional description

#### 25.4.1 XSPIM block diagram

The block diagram of the XSPI I/O manager is shown in *Figure 214*. 
25.4.2 XSPIM input/output pins

Table 217. XSPIM input/output pins

<table>
<thead>
<tr>
<th>Pin name(1)</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSPIM_Px_NCLK</td>
<td>Output</td>
<td>XSPI inverted clock to support 1.8 V HyperBus protocol</td>
</tr>
<tr>
<td>XSPIM_Px_CLK</td>
<td></td>
<td>XSPI clock</td>
</tr>
<tr>
<td>XSPIM_Px_IO[n] (n = 0 to 15)</td>
<td>Input/output</td>
<td>XSPI data pins</td>
</tr>
<tr>
<td>XSPIM_Px_NCS1,2</td>
<td>Output</td>
<td>Chip select for the memory</td>
</tr>
<tr>
<td>XSPIM_Px_DQS0,1</td>
<td>Input/output</td>
<td>Data strobe/write mask signals from/to the memory</td>
</tr>
</tbody>
</table>

1. x = 1 to 2.

25.4.3 XSPIM matrix

The XSPI I/O manager matrix allows the user to set a premapping of functions:
- XSPI1 mapped to Port 1, with XSPI2 mapped to Port 2 (direct mode)
- XSPI1 mapped to Port 2, with XSPI2 mapped to Port 1 (swapped mode)
- XSPI1 and XSPI2 both mapped to Port 1, with arbitration (multiplexed mode)
- XSPI1 and XSPI2 both mapped to Port 2, with arbitration (multiplexed mode)

Note: There is no possibility to use mixed combinations of signals (like NCS of XSPI1 with data of XSPI2).

When several I/O pins have the same configuration and are enabled at the same time, the result can be unpredictable.

In the default out-of-reset configuration, all the XSPI1 and XSPI2 signals are mapped, respectively, on Port 1 and on Port 2.

The configuration can be changed only when all XSPIs are disabled.
25.4.4 XSPIM multiplexed mode

When this mode is set, the XSPIs are time-multiplexed over the same bus. They get the ownership of the bus (in turn) through a request/acknowledge protocol with REQ/ACK signals.

The time-multiplexing is enabled by setting the MUXEN bit in the configuration register XSPIM_CR.

The fairness counter (MAXTRAN) of each XSPI can be used to manage the maximum duration for which a given XSPI takes the bus: this feature ensures the maximum bus access latency for the other XSPI(s). When one XSPI releases the bus, a round-robin arbitration phase occurs: when another XSPI requests the bus, it gets it.

When the multiplexed mode is enabled, either the fairness counter or the refresh timeout counter of both XSPI interfaces must be activated.

Each XSPI delivers one NCS signal together with a CSSEL control signal. The CSSEL signal selects which of the two outputs (NCS1, NCS2) is active. In multiplexed modes the active CCSEL signal is the one of the XSPI owning the bus. The default value of XSPI_CR.CSSEL is 0 for XSPI1 and 1 for XSPI2. Only XSPIn_IOs, XSPIn_DQS and XSPIn_CLK / XSPIn_NCLK are multiplexed.

When the multiplexed mode is used, only clock mode 0 is supported on the XSPIs.

Due to arbitration and bus sharing, the auto polling interval time of the XSPI, when used, can be increased.

**Minimum switching duration**

The minimum number of cycles needed to switch from an XSPI to another can be configured. This guarantees a latency between the falling edge of the REQ signal of the active XSPI (the active one releases the bus), and the rising edge of the ACK signal to the requesting XSPI (the bus is granted to the requesting one).

The duration is defined by the REQ2ACK_TIME field in the XSPIM_CR register.

25.5 Use cases description

The following table summarizes the use cases corresponding to IO manager modes.

<table>
<thead>
<tr>
<th>IO manager mode</th>
<th>MUXEN</th>
<th>Mode</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>0</td>
<td>0</td>
<td>Section 25.5.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Section 25.5.2</td>
</tr>
<tr>
<td>Swapped</td>
<td>0</td>
<td>1</td>
<td>Section 25.5.4</td>
</tr>
<tr>
<td>Multiplexed to Port1</td>
<td>1</td>
<td>0</td>
<td>Section 25.5.5</td>
</tr>
<tr>
<td>Multiplexed to Port2</td>
<td>1</td>
<td>1</td>
<td>Section 25.5.6</td>
</tr>
<tr>
<td>Two XSPIs drive one external memory</td>
<td>1</td>
<td>0 or 1, depending upon the port connected to the memory</td>
<td>Section 25.5.7</td>
</tr>
<tr>
<td>One XSPI drives two external memories</td>
<td>0</td>
<td>0 with CS selection command from XSPI1 or 2.</td>
<td>Section 25.5.8</td>
</tr>
</tbody>
</table>
25.5.1 XSPIs direct octal mode

This is the baseline mode: each XSPI drives independently the corresponding port and high-speed interface. There is no multiplexing.

The configuration selection signal selects the signals used for each high speed interface. These signals come from the XSPI_CR and XSPI_CALxxx register outputs.

Alternate function signals can be routed on NCS2 pins (they are not used for the XSPI bus), nclk clocks are requested in case of 1.8 V HyperBus memory type.

This mode is selected by default through bits XSPIM_CR[1:0].

25.5.2 XSPI direct 16-bit mode

This is the baseline mode, completely similar to XSPI octal direct mode, where each XSPI drives directly the corresponding port (XSPI1 connected to Port 1, XSPI2 connected to Port 2). Two DQS (1 per byte) are needed for each 16-bit port, when available.
25.5.3 **XSPI dual-octal mode**

This mode is equivalent to XSPI direct mode, but, for simplicity, only one XSPI interface is represented. Two external octal memories are accessed in parallel, similarly to the dual-quad mode implemented with octal interfaces.
25.5.4 XSPI swapped mode

This mode is similar to XSPI direct mode, but the ports are swapped, to help I/O mapping. The mode is selected by bits XSPIM.CR[1:0].

If XSPIM.CR[0] or XSPIM.CR[1] is toggled in application, XSPI DCR2 or XSPI.CCR must be written, so that the calibration of the high speed interface is launched according to the new XSPI using it.

**Note:** In swapped mode, the XSPI2 can be configured in 16-bit mode, and the XSPI1 can be configured in octal mode, to connect an external 16-bit memory on Port 1, and to connect in a concurrent way an octal external memory connected to Port 2 of the I/O manager.

**Figure 218. XSPI swapped (octal) mode**

25.5.5 Two XSPIs multiplexed mode to Port 1 accessing two external memories

If only one output port is used to access two memories, alternate function signals can be routed to pins corresponding to the other port. This mode is selected by bits XSPIM.CR[1:0].

The arbiter in IO manager selects XSPI1 or XSPI2 to own the octal-SPI bus according to the existing transfer requests and status of the two MAXTRAN fairness counters.

The external memories can be on separate chips, or embedded in a single multichip package. In this case, each memory requests a dedicated Chip select, selected according to the CSSEL control of the XSPI currently owning the bus.

The configuration signals of the high speed interface toggle when the corresponding XSPI is selected, because the timing characteristics of the two memories can be different.
25.5.6 Two XSPIs multiplexed mode to Port 2 accessing two external memories

This is similar to XSPI multiplexed mode to Port 1, but target is Port 2. This mode is selected by bits XSPIM_CR[2:1].
25.5.7 XSPI1 and XSPI2 drive a single external memory

This is similar to XSPIs multiplexed mode to Port 1, but in this case only one external memory is present. Both CSSEL controls must target the same NCS (NCS1 in Figure 221).

Figure 221. XSPI1 and XSPI2 drive a single external memory (octal mode)

This use case is valid even when Port 2 is used instead of Port 1, and also possible with XSPI1 and XSPI2 interfacing a 16-bit memory (connecting XSPIM_P1_DQS1 to the memory).

25.5.8 A single XSPI drives two external memories

This is a subset of direct mode and corresponds to semi-dynamic switching. There are two target memories, but the arbiter is not used. The software must set the XSPI1 chip selector (bit 24 of XSPI_CR register) each time the target memory changes.

The baseline use case boots on flash (memory1) and then switches to SRAM (memory2) for the rest of the application.
This use case is valid even when Port 2 is used instead of Port 1, and also with XSPI3 interfacing two 16-bit memories (connecting XSPIM_P1_DQS1 to the memory).
25.6 XSPIM registers

25.6.1 XSPIM control register (XSPIM_CR)

Address offset: 0x0000
Reset value: 0x0000 0000

| Bits 31:24 | Reserved, must be kept at reset value. |
| Bits 23:16 | **REQ2ACK_TIME[7:0]**: REQ to ACK time |
| In Multiplexed mode (MUXEN = 1), this field defines the time between two transactions. |
| The value is the number of XSPI clock cycles - 1 |
| Bits 15:7 | Reserved, must be kept at reset value. |
| Bit 6 | **CSSEL_OVR_O2**: Chip select selector override setting for XSPI2 |
| 0: the chip select signal from XSPI2 is sent to NCS1 |
| 1: the chip select signal from XSPI2 is sent to NCS2 |
| Bit 5 | **CSSEL_OVR_O1**: Chip select selector override setting for XSPI1 |
| 0: the chip select signal from XSPI1 is sent to NCS1 |
| 1: the chip select signal from XSPI1 is sent to NCS2 |
| Bit 4 | **CSSEL_OVR_EN**: Chip select selector override enable |
| 0: override not enabled |
| 1: CSSEL_OVR_O1 and CSSEL_OVR_O2 bit values overwrite the CSSEL signal values coming from XSPI1 and XSPI2 |
| Bits 3:2 | Reserved, must be kept at reset value. |
| Bit 1 | **MODE**: XSPI multiplexing mode |
| 0: if MUXEN = 0 direct mode, if MUXEN = 1 arbitration mode to output Port 1 |
| 1: if MUXEN = 0 swapped mode, if MUXEN = 1 arbitration mode to output Port 2 |
| Bit 0 | **MUXEN**: Multiplexed mode enable |
| This bit enables the multiplexing of the two XSPIs. |
| 0: No multiplexing, hence no arbitration |
| 1: XSPI1 and XSPI2 are multiplexed over the same bus |
### 25.6.2 XSPIM register map

| Offset  | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0000  | XSPIM _CR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to *Section 2.3 on page 149* for the register boundary addresses.
26 Delay block (DLYB)

26.1 Introduction
The delay block (DLYB) is used to generate an output clock that is dephased from the input clock. The phase of the output clock must be programmed by the user application. The output clock is then used to clock the data received by another peripheral such as an SDMMC interface.

The delay is voltage- and temperature-dependent, that may require the application to reconfigure and recenter the output clock phase with the receive data.

26.2 DLYB main features
The delay block has the following features:
- Input clock frequency ranging from 25 MHz to the maximum frequency supported by the communication interface (see datasheet)
- Up to 12 oversampling phases.

26.3 DLYB functional description
26.3.1 DLYB diagram
The delay block includes the following sub-blocks (shown in the figure below):
- register interface block providing AHB access to the DLYB registers
- delay line supporting the unit delays
- delay line length sampling
- output clock selection multiplexer

Figure 223. DLYB block diagram
26.3.2 DLYB pins and internal signals

Table 220 lists the DLYB internal signals.

Table 220. DLYB internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dlyb_hclk</td>
<td>Digital</td>
<td>Delay block register interface clock</td>
</tr>
<tr>
<td>dlyb_in_ck</td>
<td>Digital</td>
<td>Delay block input clock</td>
</tr>
<tr>
<td>dlyb_out_ck</td>
<td>Digital</td>
<td>Delay block output clock</td>
</tr>
</tbody>
</table>

26.3.3 General description

The delay block is enabled by setting the DEN bit in the DLYB control register (DLYB_CR). The length sampler is enabled through the SEN bit in DLYB_CR register.

When the delay block is enabled, the delay added by a unit delay is defined by the UNIT[6:0] field in the DLYB configuration register (DLYB_CFGR).

Note: UNIT[6:0] can be programmed only when the output clock is disabled (SEN = 1).

When the delay block is enabled, the output clock phase is selected through the SEL[3:0] field in DLYB_CFGR register.

Note: SEL can be programmed only when the output clock is disabled (SEN = 1).

The output clock can be de-phased over one input clock period by configuring the delay line length to span one period. The delay line length can be configured by enabling the length sampler through the SEN bit, that gives access to the delay line length (LNG[11:0]) and length valid flag (LNGF) in DLYB_CFGR.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced. This allows a smaller unit delay providing higher resolution.

Once the delay line length is configured, a dephased output clock can be selected by the output clock multiplexer. This is done through SEL[3:0]. The output clock is only available on the selected phase when SEN is set to 0.

The table below gives a summary of the delay block control.

Table 222. Delay block control

<table>
<thead>
<tr>
<th>DEN</th>
<th>SEN</th>
<th>UNIT</th>
<th>SEL</th>
<th>LNG</th>
<th>LNGF</th>
<th>Output clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Don't care</td>
<td>Enabled (= Input clock)</td>
</tr>
</tbody>
</table>
26.3.4 Delay line length configuration procedure

LNG[11:0] is used to determine the delay line length with respect to the input clock period. The length must be configured so that one full input clock period is covered by the delay line length.

Note that despite the delay line has 12 unit delay elements, the following procedure description returns a length between 0 and 10, as the upper delay output value is used to ensure that the delay is calibrated over one full input clock cycle. Depending on the clock frequency and UNIT value, unit delay element 10 may also be truncated from the clock cycle length.

A clock input (free running clock) must be present during the whole tuning procedure.

To configure the delay line length to one period of the Input clock, follow the sequence below:

1. Enable the delay block by setting DEN bit to 1.
2. Enable the length sampling by setting SEN bit to 1.
3. Enable all delay cells by setting SEL[3:0] to 12.
4. For UNIT[6:0] = 0 to 127 (this step must be repeated until the delay line length is configured):
   a) Update the UNIT[6:0] value and wait till the length flag LNGF is set to 1.
   b) Read LNG[11:0].
      If (LNG[10:0] > 0) and (LNG[11] or LNG[10] = 0), the delay line length is configured to one input clock period.
5. Determine how many unit delays (N) span one input clock period: for N = 0 to 10, if LNG[N] = 1, the number of unit delays spanning the input clock period = N.
6. Disable the length sampling by clearing SEN to 0.

If an output clock delay smaller than one input clock period is needed the delay line length can be reduced smaller than one input clock period. This allows a smaller unit delay, providing a higher resolution spanning a shorter time interval.

26.3.5 Output clock phase configuration procedure

When the delay line length is configured to one input clock period, the output clock phase can be selected between the unit delays spanning one Input clock period.

Follow the steps below to select the output clock phase:

---

### Table 222. Delay block control

<table>
<thead>
<tr>
<th>DEN</th>
<th>SEN</th>
<th>UNIT</th>
<th>SEL</th>
<th>LNG</th>
<th>LNGF</th>
<th>Output clock</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Unit delay</td>
<td>Output clock phase</td>
<td>Length</td>
<td>Length flag</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Unit delay(1)</td>
<td>Output clock phase(2)</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>Enabled (= selected phase)</td>
</tr>
</tbody>
</table>

1. The unit delay can only be changed when SEN = 1.
2. The output clock phase can only be changed when SEN = 1.
Delay block (DLYB) RM0477

1. Disable the output clock and enable the access to the phase selection SEL[3:0] bits by setting SEN bit to 1.
2. Program SEL[3:0] with the desired output clock phase value.
3. Enable the output clock on the selected phase by clearing SEN to 0.

SDMMC use case:
The delay block is used in conjunction with SDMMC interface variable delay. For correct sampling point tuning the delay value must cover a whole SDMMC_CK clock period. After having tuned the delay line length the individual delays are used in the sampling point tuning to find the optimal sampling point.

26.4 DLYB registers

All registers can be accessed in word, half-word and byte access.

26.4.1 DLYB control register (DLYB_CR)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 1 **SEN**: Sampler length enable bit
- 0: Sampler length and register access to UNIT[6:0] and SEL[3:0] disabled, output clock enabled.
- 1: Sampler length and register access to UNIT[6:0] and SEL[3:0] enabled, output clock disabled.

Bit 0 **DEN**: Delay block enable bit
- 0: DLYB disabled.
- 1: DLYB enabled.
26.4.2 DLYB configuration register (DLYB_CFRG)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LNGF</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>Res</td>
<td>LNG[11:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>UNIT[6:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 LNGF: Length valid flag
This flag indicates when the delay line length value contained in LNG[11:0] is valid after UNIT[6:0] bits changed.
0: Length value in LNG is not valid.
1: Length value in LNG is valid.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 LNG[11:0]: Delay line length value
These bits reflect the 12 unit delay values sampled at the rising edge of the input clock.
The value is only valid when LNGF = 1.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 UNIT[6:0]: Delay of a unit delay cell.
These bits can only be written when SEN = 1.
Unit delay = initial delay + UNIT[6:0] x delay step

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 SEL[3:0]: Phase for the output clock.
These bits can only be written when SEN = 1.
Output clock phase = input clock + SEL[3:0] x unit delay

26.4.3 DLYB register map

Table 223. DLYB register map and reset values

| Offset | Register name | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--------------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | DLYB_CR      | LNGF         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| Reset value | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 for the register boundary addresses.
27 Analog-to-digital converters (ADC1/2)

27.1 Introduction

This section describes the implementation of up to 2 ADCs:
- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master).

Each ADC consists of a 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 19 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned 16-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

27.2 ADC main features

- High-performance features
  - Up to 2 ADCs which can operate in dual mode:
    - ADC1 is connected to 17 external channels and to 2 internal channels
    - ADC2 is connected to 17 external channels and to 2 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Manage single-ended or differential inputs
  - AHB slave bus interface to allow fast data handling
  - Self-calibration
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Hardware assistant to prepare the context of the injected channels to allow fast context switching
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Data can be routed to ADF interface for post processing
  - Four dedicated data registers for the injected channels
• Low-power features
  – Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency
  – Allows slow bus frequency application while keeping optimum ADC performance
  – Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (auto-delayed mode)

• Oversampler
  – 16-bit data register
  – Oversampling ratio adjustable from 2 to 256x
  – Programmable data shift up to 8 bits

• Data preconditioning
  – Offset compensation

• Analog input channels
  – External analog inputs (per ADC):
    Up to 6 fast channels from GPIO pads
    Up to 11 slow channels from GPIO pads
  – 1 channel for the internal temperature sensor (VSENSE)
  – 1 channel for the internal reference voltage (VREFINT)
  – 1 channel for monitoring the external VBAT power supply pin
  – 1 channel for monitoring the internal VDDCORE supply

• Start-of-conversion can be initiated:
  – By software for both regular and injected conversions
  – By hardware triggers with configurable polarity (internal timers events or GPIO input events) for both regular and injected conversions

• Conversion modes
  – Each ADC can convert a single channel or can scan a sequence of channels
  – Single mode converts selected inputs once per trigger
  – Continuous mode converts selected inputs continuously
  – Discontinuous mode

• Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events

• 3 analog watchdogs per ADC
  – Watchdog can perform filtering to ignore out-of-range data

• ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

*Figure 224* shows the block diagram of one ADC.
27.3 ADC implementation

Table 224. ADC features

<table>
<thead>
<tr>
<th>ADC modes/features</th>
<th>ADC1</th>
<th>ADC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>12 bit</td>
<td></td>
</tr>
<tr>
<td>Maximum sampling speed</td>
<td>5 Msps (12-bit resolution)</td>
<td></td>
</tr>
<tr>
<td>Dual mode operation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hardware offset calibration</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Hardware linearity calibration</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Single-end input</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Differential input</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Injected channel conversion</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Oversampling</td>
<td>up to x256</td>
<td></td>
</tr>
<tr>
<td>Data register</td>
<td>16 bits</td>
<td></td>
</tr>
<tr>
<td>Data register FIFO depth</td>
<td>3 stages</td>
<td></td>
</tr>
<tr>
<td>DMA support</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Parallel data output to ADF</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Offset compensation</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Gain compensation</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Number of Analog watchdog</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Option register</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

Table 225. Memory location of the temperature sensor calibration values

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Memory address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TS_CAL1</td>
<td>Temperature sensor 12-bit raw data acquired by ADC1 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+} = 3.3$ V (±10 mV)</td>
<td>0x08FF F814 - 0x08FF F815</td>
</tr>
<tr>
<td>TS_CAL2</td>
<td>Temperature sensor 12-bit raw data acquired by ADC1 at 130 °C (± 5 °C), $V_{DDA} = V_{REF+} = 3.3$ V (±10 mV)</td>
<td>0x08FF F818 - 0x08FF F819</td>
</tr>
</tbody>
</table>

Table 226. Memory location of the internal reference voltage sensor calibration value

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Memory address</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFINT_CAL</td>
<td>12-bit raw data acquired by ADC1 at 30 °C (± 5 °C), $V_{DDA} = V_{REF+} = 3.3$ V (±10 mV)</td>
<td>0x08FF F810 - 0x08FF F811</td>
</tr>
</tbody>
</table>
27.4 ADC functional description

27.4.1 ADC block diagram

Figure 224 shows the ADC block diagram and Table 227 gives the ADC pin description.
27.4.2 ADC pins and internal signals

Table 227. ADC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDDA</td>
<td>Input, analog supply</td>
<td>Analog power supply and positive reference voltage for the ADC</td>
</tr>
<tr>
<td>VSSA</td>
<td>Input, analog supply ground</td>
<td>Ground for analog power supply, equal to VSS.</td>
</tr>
<tr>
<td>VREF+</td>
<td>Input, analog reference positive</td>
<td>The higher/positive reference voltage for the ADC.</td>
</tr>
<tr>
<td>VREF−</td>
<td>Input, analog reference negative</td>
<td>The lower/negative reference voltage for the ADC. VREF− is internally connected to VSSA</td>
</tr>
<tr>
<td>ADC1/2_INNi/INPi</td>
<td>Negative/positive external analog input signals</td>
<td>19 negative/positive external analog input channels (refer to Section 27.4.4: ADC connectivity for details)</td>
</tr>
</tbody>
</table>

Table 228. ADC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VINPi</td>
<td>Positive analog input channels</td>
<td>Positive internal analog input channels connected either to ADC1/2_INPi/ external channels or to internal channels.</td>
</tr>
<tr>
<td>VINNi</td>
<td>Negative analog input channels</td>
<td>Negative internal analog input channels connected either to ADC1/2_INNi/ external channels or to internal channels.</td>
</tr>
<tr>
<td>adc_ext_trgi</td>
<td>Inputs</td>
<td>ADC external trigger inputs for regular conversions. These inputs are shared between the ADC master and the ADC slave.</td>
</tr>
<tr>
<td>adc_jext_trgi</td>
<td>Inputs</td>
<td>ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.</td>
</tr>
<tr>
<td>adc_awdx</td>
<td>Output</td>
<td>Internal analog watchdog output signal connected to on-chip timers. (x = Analog watchdog number 1,2,3)</td>
</tr>
<tr>
<td>adc_ker_ck_input</td>
<td>Output</td>
<td>ADC kernel clock</td>
</tr>
<tr>
<td>adc_hclk</td>
<td>Input</td>
<td>ADC peripheral clock</td>
</tr>
<tr>
<td>adc_it</td>
<td>Output</td>
<td>ADC interrupt</td>
</tr>
<tr>
<td>adc_dma</td>
<td>Output</td>
<td>ADC DMA request</td>
</tr>
<tr>
<td>adc_dat[15:0]</td>
<td>Output</td>
<td>ADC data outputs</td>
</tr>
</tbody>
</table>

Table 229. ADC interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC1 VINP[16]</td>
<td>VSENSE (internal temperature sensor output voltage).</td>
</tr>
<tr>
<td>ADC1 VINP[17]</td>
<td>VREFINT (output voltage from internal reference voltage).</td>
</tr>
<tr>
<td>ADC2 VINP[16]</td>
<td>VBAT/4 (VBAT pin input voltage divided by 4).</td>
</tr>
<tr>
<td>ADC2 VINP[17]</td>
<td>VDDCORE (internal digital core voltage).</td>
</tr>
<tr>
<td>Signal name</td>
<td>Source/destination</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>adc_dat[15:0]</td>
<td>adf_adc1_dat[15:0], adf_adc2_dat[15:0]</td>
</tr>
<tr>
<td>adc_ext_trg0</td>
<td>tim1_oc1</td>
</tr>
<tr>
<td>adc_ext_trg1</td>
<td>tim1_oc2</td>
</tr>
<tr>
<td>adc_ext_trg2</td>
<td>tim1_oc3</td>
</tr>
<tr>
<td>adc_ext_trg3</td>
<td>tim2_oc2</td>
</tr>
<tr>
<td>adc_ext_trg4</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>adc_ext_trg5</td>
<td>tim4_oc4</td>
</tr>
<tr>
<td>adc_ext_trg6</td>
<td>exti11</td>
</tr>
<tr>
<td>adc_ext_trg7</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>adc_ext_trg8</td>
<td>tim9_trgo</td>
</tr>
<tr>
<td>adc_ext_trg9</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>adc_ext_trg10</td>
<td>tim1_trgo2</td>
</tr>
<tr>
<td>adc_ext_trg11</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>adc_ext_trg12</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>adc_ext_trg13</td>
<td>tim6_trgo</td>
</tr>
<tr>
<td>adc_ext_trg14</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>adc_ext_trg15</td>
<td>tim3_oc4</td>
</tr>
<tr>
<td>adc_ext_trg16</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg17</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg18</td>
<td>lptim1_ch1</td>
</tr>
<tr>
<td>adc_ext_trg19</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>adc_ext_trg20</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>adc_ext_trg21</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg22</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg23</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg24</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg25</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg26</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg27</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg28</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg29</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg30</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_ext_trg31</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg0</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>adc_jext_trg1</td>
<td>tim1_oc4</td>
</tr>
</tbody>
</table>
Table 229. ADC interconnection (continued)

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>adc_jext_trg2</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>adc_jext_trg3</td>
<td>tim2_oc1</td>
</tr>
<tr>
<td>adc_jext_trg4</td>
<td>tim3_oc4</td>
</tr>
<tr>
<td>adc_jext_trg5</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>adc_jext_trg6</td>
<td>exti15</td>
</tr>
<tr>
<td>adc_jext_trg7</td>
<td>tim9_oc1</td>
</tr>
<tr>
<td>adc_jext_trg8</td>
<td>tim1_trgo2</td>
</tr>
<tr>
<td>adc_jext_trg9</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>adc_jext_trg10</td>
<td>tim9_trgo</td>
</tr>
<tr>
<td>adc_jext_trg11</td>
<td>tim3_oc3</td>
</tr>
<tr>
<td>adc_jext_trg12</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>adc_jext_trg13</td>
<td>tim3_oc1</td>
</tr>
<tr>
<td>adc_jext_trg14</td>
<td>tim6_trgo</td>
</tr>
<tr>
<td>adc_jext_trg15</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>adc_jext_trg16</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg17</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg18</td>
<td>lptim1_ch2</td>
</tr>
<tr>
<td>adc_jext_trg19</td>
<td>lptim2_ch2</td>
</tr>
<tr>
<td>adc_jext_trg20</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>adc_jext_trg21</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg22</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg23</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg24</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg25</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg26</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg27</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg28</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg29</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg30</td>
<td>reserved</td>
</tr>
<tr>
<td>adc_jext_trg31</td>
<td>reserved</td>
</tr>
</tbody>
</table>
27.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC clock is independent from the AHB bus clock.

The ADC input clock can be selected between two different clock sources (see Figure 225: ADC clock scheme):

1. The ADC clock can be a specific clock source (adc_ker_ck_input), independent and asynchronous with the AHB clock.
   Refer to section Reset and clock control (RCC) for more information on how to generate the ADC dedicated clock. To select this scheme, CKMODE[1:0] bits of ADC_CCR register must be set to 00.

2. The ADC clock can be derived from the AHB clock interface divided by a programmable factor of 1, 2 or 4. To select this scheme, CKMODE[1:0] bits of ADC_CCR must be different from 00. The programmable divider factor can be configured through to CKMODE[1:0] bits of ADC_CCR.
   The prescaling factor of 1 (CKMODE[1:0] = 01) can be used only if the AHB prescaler is set to 1 (HPRE[3:0] = 0xx in the RCC_CFGR register).

Option 1 has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected. The ADC clock can eventually be divided by the following ratio: 1, 2, 4, 6, 8, 12, 16, 32, 64, 128, 256, using the prescaler configured with bits PRESC[3:0] in the ADC_CCR register.

Option 2 has the advantage of bypassing the clock domain resynchronizations. This can be useful when the ADC is triggered by a timer and if the application requires that the ADC is precisely triggered without any uncertainty (otherwise, an uncertainty of the trigger instant is added by the resynchronizations between the two clock domains).

The clock is configured through CKMODE[1:0] bits must be compliant with the operating frequency specified in the device datasheet.
Clock ratio constraint between ADC clock and AHB clock

There are generally no constraints to be respected for the ratio between the ADC clock and the AHB clock except if some injected channels are programmed. In this case, it is mandatory to respect the following ratio:

- $F_{\text{adc\_hclk}} \geq F_{\text{ADC}} / 4$ if the resolution of all channels are 12-bit or 10-bit
- $F_{\text{adc\_hclk}} \geq F_{\text{ADC}} / 3$ if there are some channels with resolutions equal to 8-bit (and none with lower resolutions)
- $F_{\text{adc\_hclk}} \geq F_{\text{ADC}} / 2$ if there are some channels with resolutions equal to 6-bit

Constraints between ADC clocks

When several ADC interfaces are used simultaneously, it is mandatory to use the same clock source from the RCC block without prescaler ratio for all ADC interfaces.
27.4.4 ADC connectivity

ADC inputs are connected to the external channels as well as internal sources as described below.

Figure 226. ADC1 connectivity
Figure 227. ADC2 connectivity

- ADC2_INP0
- ADC2_INP1
- ADC2_INP2
- ADC2_INP3
- ADC2_INP4
- ADC2_INP5
- ADC2_INP6
- ADC2_INN2
- ADC2_INN3
- ADC2_INN4
- ADC2_INN5
- ADC2_INN6
- ADC2_INN7
- ADC2_INN8
- ADC2_INP9
- ADC2_INP10
- ADC2_INN10
- ADC2_INP11
- ADC2_INN11
- ADC2_INP12
- ADC2_INP13
- ADC2_INN13
- ADC2_INP14
- ADC2_INN14
- ADC2_INP15
- ADC2_INN15
- ADC2_INP16
- ADC2_INN16
- ADC2_INP17
- ADC2_INN17
- ADC2_INP18
- ADC2_INN18

Channel selection

- VINP[0]
- VIN[0] Fast channel
- VINP[1]
- VIN[1] Fast channel
- VINP[2]
- VIN[2] Fast channel
- VINP[3]
- VIN[3] Fast channel
- VINP[4]
- VIN[4] Fast channel
- VINP[5]
- VIN[5] Fast channel
- VINP[6]
- VIN[6] Slow channel
- VINP[7]
- VIN[7] Slow channel
- VINP[8]
- VIN[8] Slow channel
- VINP[9]
- VIN[9] Slow channel
- VINP[10]
- VIN[10] Slow channel
- VINP[11]
- VINP[12]
- VIN[12] Slow channel
- VINP[13]
- VIN[13] Slow channel
- VINP[14]
- VIN[14] Slow channel
- VINP[15]
- VIN[15] Slow channel
- VINP[16]
- VIN[16] Slow channel
- VINP[17]
- VIN[17] Slow channel
- VINP[18]
- VIN[18] Slow channel

VREF+

VDDCORE

VSSA

VREF−
27.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word (32-bit) accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and never generates AHB errors.

27.4.6 ADC Deep-power-down mode (DEEPWD) and ADC voltage regulator (ADVREGEN)

By default, the ADC is in Deep-power-down mode where its supply is internally switched off to reduce the leakage currents (the reset state of bit DEEPWD is 1 in the ADC_CR register).

To start ADC operations, it is first needed to exit Deep-power-down mode by setting bit DEEPWD = 0.

Then, it is mandatory to enable the ADC internal voltage regulator by setting the bit ADVREGEN = 1 into ADC_CR register. The software must wait for the startup time of the ADC voltage regulator \( T_{ADVREG\_STUP} \) before launching a calibration or enabling the ADC. This delay must be implemented by software.

For the startup time of the ADC voltage regulator, refer to device datasheet for \( T_{ADVREG\_STUP} \) parameter.

When ADC operations are complete, the ADC can be disabled (ADEN = 0). It is possible to save power by also disabling the ADC voltage regulator. This is done by writing bit ADVREGEN = 0.

Then, to save more power by reducing the leakage currents, it is also possible to re-enter in ADC Deep-power-down mode by setting bit DEEPWD = 1 into ADC_CR register. This is particularly interesting before entering Stop mode.

**Note:** *Writing DEEPWD = 1 automatically disables the ADC voltage regulator and bit ADVREGEN is automatically cleared.*

*When the internal voltage regulator is disabled (ADVREGEN = 0), the internal analog calibration is kept.*

In ADC Deep-power-down mode (DEEPWD = 1), the internal analog calibration is lost and it is necessary to either relaunch a calibration or re-apply the calibration factor which was previously saved (refer to Section 27.4.8: Calibration (ADCAL, ADCALDIF, ADC_CALFACT)).
27.4.7 Single-ended and differential input channels

Channels can be configured to be either single-ended input or differential input by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be written while the ADC is disabled (ADEN = 0). Note that the DIFSEL[i] bits corresponding to single-ended channels are always programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage \( V_{\text{INP}[i]} \) (positive input) and \( V_{\text{REF}} \).

In differential input mode, the analog voltage to be converted for channel “i” is the difference between the external voltage \( V_{\text{INP}[i]} \) (positive input) and \( V_{\text{INN}[i]} \) (negative input).

The output data for the differential mode is an unsigned data. When \( V_{\text{INP}[i]} \) equals \( V_{\text{REF}} \), \( V_{\text{INN}[i]} \) equals \( V_{\text{REF}} \) and the output data is 0x000 (12-bit resolution mode). When \( V_{\text{INP}[i]} \) equals \( V_{\text{REF}} \), \( V_{\text{INN}[i]} \) equals \( V_{\text{REF}} \) and the output data is 0xFFF.

\[
\text{Converted value} = \frac{\text{ADC Full Scale}}{2} \times \left( 1 + \frac{V_{\text{INP}} - V_{\text{INN}}}{V_{\text{REF}}} \right)
\]

When ADC is configured as differential mode, both inputs should be biased at \((V_{\text{REF}})/2\) voltage.

The input signals are supposed to be differential (common mode voltage should be fixed).

Internal channels (such as \( V_{\text{REFINT}} \) and \( V_{\text{SENSE}} \)) are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to Section 27.4.4: ADC connectivity.

Caution: When configuring the channel “i” in differential input mode, its negative input voltage \( V_{\text{INN}[i]} \) is connected to another channel. As a consequence, this channel is no longer usable in single-ended mode or in differential mode and must never be configured to be converted. Some channels are shared between ADC1/ADC2: this can make the channel on the other ADC unusable. Only exception is interleaved mode for ADC master and the slave.

27.4.8 Calibration (ADCAL, ADCALDIFF, ADC_CALFACT)

Each ADC provides an automatic calibration procedure which drives all the calibration sequence including the power-on/off sequence of the ADC. During the procedure, the ADC calculates a calibration factor which is 7-bit wide and which is applied internally to the ADC until the next ADC power-off. During the calibration procedure, the application must not use the ADC and must wait until calibration is complete.

Calibration is preliminary to any ADC operation. It removes the offset error which may vary from chip to chip due to process or bandgap variation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Write ADCALDIFF = 0 before launching a calibration which is applied for single-ended input conversions.
- Write ADCALDIFF = 1 before launching a calibration which is applied for differential input conversions.

The calibration is then initiated by software by setting bit ADCAL = 1. Calibration can only be initiated when the ADC is disabled (when ADEN = 0). ADCAL bit stays at 1 during all the
calibration sequence. It is then cleared by hardware as soon the calibration completes. At this time, the associated calibration factor is stored internally in the analog ADC and also in the bits CALFACT_S[6:0] or CALFACT_D[6:0] of ADC_CALFACT register (depending on single-ended or differential input calibration).

The internal analog calibration is kept if the ADC is disabled (ADEN = 0). However, if the ADC is disabled for extended periods, then it is recommended that a new calibration cycle is run before re-enabling the ADC.

The internal analog calibration is lost each time the power of the ADC is removed (example, when the product enters in Standby or VBAT mode). In this case, to avoid spending time recalibrating the ADC, it is possible to re-write the calibration factor into the ADC_CALFACT register without recalibrating, supposing that the software has previously saved the calibration factor delivered during the previous calibration.

The calibration factor can be written if the ADC is enabled but not converting (ADEN = 1 and ADSTART = 0 and JADSTART = 0). Then, at the next start of conversion, the calibration factor is automatically injected into the analog ADC. This loading is transparent and does not add any cycle latency to the start of the conversion. It is recommended to recalibrate when VREF+ voltage changed more than 10%.

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADVREGEN = 1 and that ADC voltage regulator startup time has elapsed.
2. Ensure that ADEN = 0.
3. Select the input mode for this calibration by setting ADCALDIF = 0 (single-ended input) or ADCALDIF = 1 (differential input).
4. Set ADCAL = 1.
5. Wait until ADCAL = 0.
6. The calibration factor can be read from ADC_CALFACT register.

Figure 228. ADC calibration
Software procedure to reinject a calibration factor into the ADC

1. Ensure ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).
2. Write CALFACT_S and CALFACT_D with the new calibration factors.
3. When a conversion is launched, the calibration factor is injected into the analog ADC only if the internal analog calibration factor differs from the one stored in bits CALFACT_S for single-ended input channel or bits CALFACT_D for differential input channel.

Converting single-ended and differential analog inputs with a single ADC

If the ADC is supposed to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1. The procedure is the following:

1. Disable the ADC.
2. Calibrate the ADC in single-ended input mode (with ADCALDIF = 0). This updates the register CALFACT_S[6:0].
3. Calibrate the ADC in differential input modes (with ADCALDIF = 1). This updates the register CALFACT_D[6:0].
4. Enable the ADC, configure the channels and launch the conversions. Each time there is a switch from a single-ended to a differential inputs channel (and vice-versa), the calibration is automatically injected into the analog ADC.
27.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First of all, follow the procedure explained in Section 27.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN)). Once DEEPPWD = 0 and ADVREGEN = 1, the ADC can be enabled and the ADC needs a stabilization time of $t_{\text{STAB}}$ before it starts converting accurately, as shown in Figure 231. Two control bits enable or disable the ADC:

- ADEN = 1 enables the ADC. The flag ADRDY is set once the ADC is ready for operation.
- ADDIS = 1 disables the ADC. ADEN and ADDIS are then automatically cleared by hardware as soon as the analog ADC is effectively disabled.

Regular conversion can then start either by setting ADSTART = 1 (refer to Section 27.4.18: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)) or when an external trigger event occurs, if triggers are enabled.

Injected conversions start by setting JADSTART = 1 or when an external injected trigger event occurs, if injected triggers are enabled.

**Software procedure to enable the ADC**

1. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’.
2. Set ADEN = 1.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done using the associated interrupt (setting ADRDYIE = 1).
4. Clear the ADRDY bit in the ADC_ISR register by writing ‘1’ (optional).

**Caution:** ADEN bit cannot be set when ADCAL is set and during four ADC clock cycles after the ADCAL bit is cleared by hardware (end of the calibration).
Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop any regular and injected conversion ongoing by setting ADSTP = 1 and JADSTP = 1 and then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS = 1.
3. If required by the application, wait until ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 231. Enabling / disabling the ADC

27.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC Section), the DIFSEL[i] control bits in the ADC_DIFSEL register and the control bits ADCAL and ADEN in the ADC_CR register, only if the ADC is disabled (ADEN must be equal to 0).

The software is then allowed to write the control bits ADSTART, JADSTART and ADDIS of the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

For all the other control bits of the ADC_CFGR, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no regular conversion ongoing (ADSTART must be equal to 0).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if the ADC is enabled (ADEN = 1) and if there is no injected conversion ongoing (JADSTART must be equal to 0).
- ADC_TRy registers can be modified when an analog-to-digital conversion is ongoing (refer to Section 27.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx) for details).

The software is allowed to write the ADSTP or JADSTP control bits of the ADC_CR register only if the ADC is enabled, possibly converting, and if there is no pending request to disable the ADC (ADSTART or JADSTART must be equal to 1 and ADDIS to 0).
The software can write the register ADC_JSQR at any time, when the ADC is enabled (ADEN = 1). Refer to Section 27.7.16: ADC injected sequence register (ADC_JSQR) for additional details.

**Note:** There is no hardware protection to prevent these forbidden write accesses and ADC behavior may become in an unknown state. To recover from this situation, the ADC must be disabled (clear ADEN = 0 as well as all the bits of ADC_CR register).

### 27.4.11 Channel selection (SQRx, JSQRx)

The ADC features up to 19 multiplexed channels per ADC, out of which:

- Up to 17 analog inputs coming from GPIO pads (ADC_INP/INN[i]) depending on the products, not all of them are available on GPIO pads.
- ADC is connected to 4 internal analog inputs:
  - the internal temperature sensor \((V_{\text{SENSE}})\)
  - the internal reference voltage \((V_{\text{REFINT}})\)
  - the \(V_{\text{BAT}}\) monitoring channel \((V_{\text{BAT}}/4)\)
  - the internal digital core voltage \((V_{\text{DDCORE}})\)

To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming bits VREFEN, VBATEN or TSEN in the ADC_CCR registers.

Refer to Table ADC interconnection in Section 27.4.2: ADC pins and internal signals for the connection of the above internal analog inputs to external ADC pins or internal signals.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADC1/2_INP/INN3, ADC1/2_INP/INN8, ADC1/2_INP/INN2, ADC1/2_INN/INP2, ADC1/2_INP/INN0, ADC1/2_INP/INN2, ADC1/2_INP/INN15.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the L[3:0] bits in the ADC_SQR1 register.

- An **injected group** is composed of up to 4 conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the L[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while regular conversions can occur. For this, the ADC regular conversions must be first stopped by writing ADSTP = 1 (refer to Section 27.4.17: Stopping an ongoing conversion (ADSTP, JADSTP)).

The software is allowed to modify on-the-fly the ADC_JSQR register when JADSTART is set to 1 (injected conversions ongoing) only when the context queue is enabled (JQDIS = 0 in ADC_CFGR register). Refer to Section 27.4.21: Queue of context for injected conversions.
27.4.12 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be enough for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time which is programmable using the SMP[2:0] bits in the ADC_SMPR1 and ADC registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 2.5 ADC clock cycles
- SMP = 001: 6.5 ADC clock cycles
- SMP = 010: 12.5 ADC clock cycles
- SMP = 011: 24.5 ADC clock cycles
- SMP = 100: 47.5 ADC clock cycles
- SMP = 101: 92.5 ADC clock cycles
- SMP = 110: 247.5 ADC clock cycles
- SMP = 111: 640.5 ADC clock cycles

The total conversion time is calculated as follows:

\[ T_{\text{CONV}} = \text{Sampling time} + 12.5 \text{ ADC clock cycles} \]

Example:

With \( F_{\text{adc, ker, ck}} = 30 \text{ MHz} \) and a sampling time of 2.5 ADC clock cycles:

\[ T_{\text{CONV}} = (2.5 + 12.5) \text{ ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns} \]

The ADC notifies the end of the sampling phase by setting the status bit EOSMP (only for regular conversion).

Note: Depending on the ADC conversion mode, the real sampling time can vary compared to the SMP value programmed above, while the equivalent total conversion time \( T_{\text{CONV}} \) does not change:

- For the first conversion in scan or continuous mode and all the conversions in discontinuous mode, the real sampling time is 0.5 clock cycle less compared to the value configured above.
- For the second and subsequent conversions in scan or continuous mode, 0.5 cycle is added to the configured sampling time. This additional 0.5 clock cycle overlaps with the previous conversion cycle.

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum sampling time as specified in the ADC characteristics section of the datasheets.
Bulb sampling mode
When the BULB bit is set in ADC register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPR1 register. The very first ADC conversion, after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

The bulb mode is neither compatible with the continuous conversion mode nor with the injected channel conversion.

When the BULB bit is set, it is not allowed to set SMPTRIG bit in ADC_CFGR2.

Sampling time control trigger mode
When the SMPTRIG bit is set, the sampling time programmed though SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected, each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

When a software trigger is selected, the software trigger is not the ADSTART bit in ADC_CR but the SWTRIG bit. SWTRIG bit has to be set to start the sampling period, and the SWTRIG bit has to be cleared to end the sampling period and start the conversion.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is neither compatible with the continuous conversion mode, nor with the injected channel conversion.

When SMPTRIG bit is set, it is not allowed to set BULB bit.

I/O analog switch voltage booster
The resistance of the I/O analog switches increases when the $V_{DDA}$ voltage is too low. The sampling time must consequently be adapted accordingly (refer to the device datasheet for the corresponding electrical characteristics). This resistance can be minimized at low $V_{DDA}$
by enabling an internal voltage booster through BOOSTE bit or by selecting a VDD booster voltage (if VDD > 2.7 V) through the ADV_READY bit of the PWR_PMCR register.

**SMPLPLUS control bit**

The SMPLPLUS bit can be used to change the sampling time from 2.5 ADC clock cycles to 3.5 ADC clock cycles.

### 27.4.13 Single conversion mode (CONT = 0)

In single conversion mode, the ADC performs once all the conversions of the channels. This mode is started with the CONT bit at 0 by either:
- Setting the ADSTART bit in the ADC_CR register (for a regular channel)
- Setting the JADSTART bit in the ADC_CR register (for an injected channel)
- External hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:
- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of regular conversion) flag is set
- An interrupt is generated if the EOCIE bit is set

Inside the injected sequence, after each conversion is complete:
- The converted data are stored into one of the four 16-bit ADC_JDRy registers
- The JEOC (end of injected conversion) flag is set
- An interrupt is generated if the JEOCIE bit is set

After the regular sequence is complete:
- The EOS (end of regular sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

After the injected sequence is complete:
- The JEOS (end of injected sequence) flag is set
- An interrupt is generated if the JEOSIE bit is set

Then the ADC stops until a new external regular or injected trigger occurs or until bit ADSTART or JADSTART is set again.

*Note:* To convert a single channel, program a sequence with a length of 1.

### 27.4.14 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts each conversions of the sequence. This mode is started with the CONT bit at 1 either by external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:
- The converted data are stored into the 16-bit ADC_DR register
- The EOC (end of conversion) flag is set
- An interrupt is generated if the EOCIE bit is set
After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set
- An interrupt is generated if the EOSIE bit is set

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

**Note:** To convert a single channel, program a sequence with a length of 1. It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit), refer to Auto-injection mode section.

### 27.4.15 Starting conversions (ADSTART, JADSTART)

Software starts ADC regular conversions by setting ADSTART = 1.

When ADSTART is set, the conversion starts:

- Immediately: if EXTEN = 0x0 (software trigger)
- At the next active edge of the selected regular hardware trigger: if EXTEN is not equal to 0x0

Software starts ADC injected conversions by setting JADSTART = 1.

When JADSTART is set, the conversion starts:

- Immediately, if JEXTEN = 0x0 (software trigger)
- At the next active edge of the selected injected hardware trigger: if JEXTEN is not equal to 0x0

**Note:** In auto-injection mode (JAUTO = 1), use ADSTART bit to start the regular conversions followed by the auto-injected conversions (JADSTART must be kept cleared).

ADSTART and JADSTART also provide information on whether any ADC operation is currently ongoing. It is possible to re-configure the ADC while ADSTART = 0 and JADSTART = 0 are both true, indicating that the ADC is idle.

ADSTART is cleared by hardware:

- In single mode with software regular trigger (CONT = 0, EXTSEL = 0x0)
  - At any end of regular conversion sequence (EOS assertion) or at any end of subgroup processing if DISCEN = 1
- In all cases (CONT = x, EXTSEL = x)
  - After execution of the ADSTP procedure asserted by the software.

**Note:** In continuous mode (CONT = 1), ADSTART is not cleared by hardware with the assertion of EOS because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTSEL ≠ 0x00), ADSTART is not cleared by hardware with the assertion of EOS to help the software which does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.
JADSTART is cleared by hardware:

- In single mode with software injected trigger (JEXTSEL = 0x0)
  - At any end of injected conversion sequence (JEOS assertion) or at any end of subgroup processing if JDISCEN = 1
- In all cases (JEXTSEL = x)
  - After execution of the JADSTP procedure asserted by the software.

Note: When the software trigger is selected, ADSTART bit should not be set if the EOC flag is still high.

27.4.16 ADC timing

The elapsed time between the start of a conversion and the end of conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

\[
T_{CONV} = T_{SMPL} + T_{SAR} = [2.5 \text{ min} + 12.5 \text{ 12bit}] \times T_{ADC\_CLK}
\]

\[
T_{CONV} = T_{SMPL} + T_{SAR} = 83.33 \text{ ns min} + 416.67 \text{ ns 12bit} = 500.0 \text{ ns (for F}_{ADC\_CLK} = 30 \text{ MHz)}
\]

Figure 233. Analog-to-digital conversion time

1. \(T_{SMPL}\) depends on SMP[2:0].
2. \(T_{SAR}\) depends on RES[2:0].
27.4.17 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop regular conversions ongoing by setting ADSTP = 1 and injected conversions ongoing by setting JADSTP = 1.

Stopping conversions resets the ongoing ADC operation. Then the ADC can be reconfigured (ex: changing the channel selection or the trigger) ready for a new operation.

Note that it is possible to stop injected conversions while regular conversions are still operating and vice-versa. This allows, for instance, re-configuration of the injected conversion sequence and triggers while regular conversions are still operating (and vice-versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with partial result discarded (ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC would restart a new sequence).

Once this procedure is complete, bits ADSTP/ADSTART (in case of regular conversion), or JADSTP/JADSTART (in case of injected conversion) are cleared by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 234. Stopping ongoing regular conversions
27.4.18 Conversion on external trigger and trigger polarity
(EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (such as timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

When the Injected Queue is enabled (bit JQDIS = 0), injected software triggers are not possible.

The regular trigger selection is effective once software has set bit ADSTART = 1 and the injected trigger selection is effective once software has set bit JADSTART = 1.

Any hardware triggers which occur while a conversion is ongoing are ignored.
- If bit ADSTART = 0, any regular hardware triggers which occur are ignored.
- If bit JADSTART = 0, any injected hardware triggers which occur are ignored.

Table 230 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

### Table 230. Configuring the trigger polarity for regular external triggers

<table>
<thead>
<tr>
<th>EXTEN[1:0]</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>Hardware Trigger detection disabled, software trigger detection enabled</td>
</tr>
<tr>
<td>01</td>
<td>Hardware Trigger with detection on the rising edge</td>
</tr>
<tr>
<td>10</td>
<td>Hardware Trigger with detection on the falling edge</td>
</tr>
<tr>
<td>11</td>
<td>Hardware Trigger with detection on both the rising and falling edges</td>
</tr>
</tbody>
</table>

**Note:** The polarity of the regular trigger cannot be changed on-the-fly.
The polarity of the injected trigger can be anticipated and changed on-the-fly when the queue is enabled (JQDIS = 0). Refer to Section 27.4.21: Queue of context for injected conversions.

The EXTSEL and JEXTSEL control bits select which out of 32 possible events can trigger conversion for the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Note: The regular trigger selection cannot be changed on-the-fly. The injected trigger selection can be anticipated and changed on-the-fly. Refer to Section 27.4.21: Queue of context for injected conversions on page 1106.

### Figure 236. Triggers shared between ADC master and slave
Refer to Table ADC interconnection in Section 27.4.2: ADC pins and internal signals for the list of all the external triggers that can be used for regular conversion.

27.4.19 Injected channel management

Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once).

3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.

4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. Figure 237 shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence. For instance, if the sequence length is 30 ADC clock cycles (that is two conversions with a sampling time of 2.5 clock periods), the minimum interval between triggers must be 31 ADC clock cycles.

Auto-injection mode

If the JAUTO bit in the ADC_CFGR register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

If the CONT bit is also set in addition to the JAUTO bit, regular channels followed by injected channels are continuously converted.

Note: It is not possible to use both the auto-injected and discontinuous modes simultaneously.

When the DMA is used for exporting regular sequencer’s data in JAUTO mode, it is necessary to program it in circular mode (CIRC bit set in DMA_CCRx register). If the CIRC bit is reset (single-shot mode), the JAUTO sequence is stopped upon DMA Transfer Complete event.
27.4.20 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN bit in the ADC_CFGR register.

It is used to convert a short sequence (subgroup) of n conversions \((n \leq 8)\) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of \(n\) is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR register.

When an external trigger occurs, it starts the next \(n\) conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the L[3:0] bits in the ADC_SQR1 register.

Example:
- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - 2nd trigger: channels converted are 6, 7, 8 (an EOC event is generated at each conversion).
  - 3rd trigger: channels converted are 9, 10, 11 (an EOC event is generated at each conversion) and an EOS event is generated after the conversion of channel 11.
  - 4th trigger: channels converted are 1, 2, 3 (an EOC event is generated at each conversion).
  - ...

- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - 1st trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.
Analog-to-digital converters (ADC1/2) RM0477

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the 4th trigger reconverts the channels 1, 2 and 3 in the 1st subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels where ‘n’ is fixed to 1.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JL[1:0] bits in the ADC_JSQR register.

Example:
- JDISCEN = 1, channels to be converted = 1, 2, 3
  - 1st trigger: channel 1 converted (a JEOC event is generated)
  - 2nd trigger: channel 2 converted (a JEOC event is generated)
  - 3rd trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all microcontrollers.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the 4th trigger reconverts the 1st injected channel 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

27.4.21 Queue of context for injected conversions

A queue of context is implemented to anticipate up to 2 contexts for the next injected sequence of conversions. JQDIS bit of ADC_CFGR register must be reset to enable this feature. Only hardware-triggered conversions are possible when the context queue is enabled.

This context consists of:
- Configuration of the injected triggers (bits JEXTEN[1:0] and JEXTSEL bits in ADC_JSQR register)
- Definition of the injected sequence (bits JSQx[4:0] and JL[1:0] in ADC_JSQR register)
All the parameters of the context are defined into a single register ADC_JSQR and this register implements a queue of 2 buffers, allowing the bufferization of up to 2 sets of parameters:

- The JSQR register can be written at any moment even when injected conversions are ongoing.
- Each data written into the JSQR register is stored into the Queue of context.
- At the beginning, the Queue is empty and the first write access into the JSQR register immediately changes the context and the ADC is ready to receive injected triggers.
- Once an injected sequence is complete, the Queue is consumed and the context changes according to the next JSQR parameters stored in the Queue. This new context is applied for the next injected sequence of conversions.
- A Queue overflow occurs when writing into register JSQR while the Queue is full. This overflow is signaled by the assertion of the flag JQOVF. When an overflow occurs, the write access of JSQR register which has created the overflow is ignored and the queue of context is unchanged. An interrupt can be generated if bit JQOVFIE is set.
- Two possible behaviors are possible when the Queue becomes empty, depending on the value of the control bit JQM of register ADC_CFGR:
  - If JQM = 0, the Queue is empty just after enabling the ADC, but then it can never be empty during run operations: the Queue always maintains the last active context and any further valid start of injected sequence is served according to the last active context.
  - If JQM = 1, the Queue can be empty after the end of an injected sequence or if the Queue is flushed. When this occurs, there is no more context in the queue and hardware triggers are disabled. Therefore, any further hardware injected triggers are ignored until the software re-writes a new injected context into JSQR register.
- Reading JSQR register returns the current JSQR context which is active at that moment. When the JSQR context is empty, JSQR is read as 0x0000.
- The Queue is flushed when stopping injected conversions by setting JADSTP = 1 or when disabling the ADC by setting ADDIS = 1:
  - If JQM = 0, the Queue is maintained with the last active context.
  - If JQM = 1, the Queue becomes empty and triggers are ignored.

Note: When configured in discontinuous mode (bit JDISCEN = 1), only the last trigger of the injected sequence changes the context and consumes the Queue. The 1st trigger only consumes the queue but others are still valid triggers as shown by the discontinuous mode example below (length = 3 for both contexts):

- 1st trigger, discontinuous. Sequence 1: context 1 consumed, 1st conversion carried out
- 2nd trigger, disc. Sequence 1: 2nd conversion.
- 3rd trigger, discontinuous. Sequence 1: 3rd conversion.
- 4th trigger, discontinuous. Sequence 2: context 2 consumed, 1st conversion carried out.
- 5th trigger, discontinuous. Sequence 2: 2nd conversion.
- 6th trigger, discontinuous. Sequence 2: 3rd conversion.
Behavior when changing the trigger or sequence context

*Figure 238* and *Figure 239* show the behavior of the context Queue when changing the sequence or the triggers.

**Figure 238. Example of JSQR queue of context (sequence change)**

1. Parameters:
   - P1: sequence of 3 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 4 conversions, hardware trigger 1

**Figure 239. Example of JSQR queue of context (trigger change)**

1. Parameters:
   - P1: sequence of 2 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 2
   - P3: sequence of 4 conversions, hardware trigger 1
Queue of context: Behavior when a queue overflow occurs

The Figure 240 and Figure 241 show the behavior of the context Queue if an overflow occurs before or during a conversion.

**Figure 240. Example of JSQR queue of context with overflow before conversion**

1. Parameters:
   - P1: sequence of 2 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 2
   - P3: sequence of 3 conversions, hardware trigger 1
   - P4: sequence of 4 conversions, hardware trigger 1

**Figure 241. Example of JSQR queue of context with overflow during conversion**

1. Parameters:
   - P1: sequence of 2 conversions, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 2
   - P3: sequence of 3 conversions, hardware trigger 1
   - P4: sequence of 4 conversions, hardware trigger 1
It is recommended to manage the queue overflows as described below:

- After each P context write into JSQR register, flag JQOVF shows if the write has been ignored or not (an interrupt can be generated).
- Avoid Queue overflows by writing the third context (P3) only once the flag JEOS of the previous context P2 has been set. This ensures that the previous context has been consumed and that the queue is not full.

**Queue of context: Behavior when the queue becomes empty**

*Figure 242* and *Figure 243* show the behavior of the context Queue when the Queue becomes empty in both cases JQM = 0 or 1.

**Figure 242. Example of JSQR queue of context with empty queue (case JQM = 0)**

1. Parameters:
   - P1: sequence of 1 conversion, hardware trigger 1
   - P2: sequence of 1 conversion, hardware trigger 1
   - P3: sequence of 1 conversion, hardware trigger 1

   **Note:** When writing P3, the context changes immediately. However, because of internal resynchronization, there is a latency and if a trigger occurs just after or before writing P3, it can happen that the conversion is launched considering the context P2. To avoid this situation, the user must ensure that there is no ADC trigger happening when writing a new context that applies immediately.
Figure 243. Example of JSQR queue of context with empty queue (JQM = 1)

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Flush the queue of context

The figures below show the behavior of the context Queue in various situations when the queue is flushed.

Figure 244. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion.

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 245. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs during an ongoing conversion and a new trigger occurs

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1

Figure 246. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 0) - JADSTP occurs outside an ongoing conversion

1. Parameters:
P1: sequence of 1 conversion, hardware trigger 1
P2: sequence of 1 conversion, hardware trigger 1
P3: sequence of 1 conversion, hardware trigger 1
Figure 247. Flushing JSQR queue of context by setting JADSTP = 1 (JQM = 1)

<table>
<thead>
<tr>
<th>Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: sequence of 1 conversion, hardware trigger 1</td>
</tr>
<tr>
<td>P2: sequence of 1 conversion, hardware trigger 1</td>
</tr>
<tr>
<td>P3: sequence of 1 conversion, hardware trigger 1</td>
</tr>
</tbody>
</table>

Figure 248. Flushing JSQR queue of context by setting ADDIS = 1 (JQM = 0)

<table>
<thead>
<tr>
<th>Parameters:</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: sequence of 1 conversion, hardware trigger 1</td>
</tr>
<tr>
<td>P2: sequence of 1 conversion, hardware trigger 1</td>
</tr>
<tr>
<td>P3: sequence of 1 conversion, hardware trigger 1</td>
</tr>
</tbody>
</table>
Queue of context: Starting the ADC with an empty queue

The following procedure must be followed to start ADC operation with an empty queue, in case the first context is not known at the time the ADC is initialized. This procedure is only applicable when JQM bit is reset:

5. Write a dummy JSQR with JEXTEN not equal to 0 (otherwise triggering a software conversion)
6. Set JADSTART
7. Set JADSTP
8. Wait until JADSTART is reset
9. Set JADSTART.

Disabling the queue

It is possible to disable the queue by setting bit JQDIS = 1 into the ADC_CFGR register.

27.4.22 Programmable resolution (RES) - fast conversion mode

It is possible to perform faster conversion by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the control bits RES[1:0]. Figure 254, Figure 255, Figure 256 and Figure 257 show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to Table 232.
### 27.4.23 End of conversion, end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application for each end of regular conversion (EOC) event and each injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if bit EOCIE is set. EOC flag is cleared by the software either by writing 1 to it or by reading ADC_DR.

The ADC sets the JEOC flag as soon as a new injected conversion data is available in one of the ADC_JDRy register. An interrupt can be generated if bit JEOCIE is set. JEOC flag is cleared by the software either by writing 1 to it or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of Sampling phase by setting the status bit EOSMP (for regular conversions only). EOSMP flag is cleared by software by writing 1 to it. An interrupt can be generated if bit EOSMPIE is set.

### 27.4.24 End of conversion sequence (EOS, JEOS)

The ADC notifies the application for each end of regular sequence (EOS) and for each end of injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the last data of the regular conversion sequence is available in the ADC_DR register. An interrupt can be generated if bit EOSIE is set. EOS flag is cleared by the software either by writing 1 to it.

The ADC sets the JEOS flag as soon as the last data of the injected conversion sequence is complete. An interrupt can be generated if bit JEOSIE is set. JEOS flag is cleared by the software either by writing 1 to it.

---

#### Table 232. $T_{\text{SAR}}$ timings depending on resolution

<table>
<thead>
<tr>
<th>RES (bits)</th>
<th>$T_{\text{SAR}}$ (ADC clock cycles)</th>
<th>$T_{\text{SAR}}$ (ns) at $F_{\text{ADC}} = 30$ MHz</th>
<th>$T_{\text{CONV}}$ (ADC clock cycles) (with Sampling Time = 2.5 ADC clock cycles)</th>
<th>$T_{\text{CONV}}$ (ns) at $F_{\text{ADC}} = 30$ MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>12.5 ADC clock cycles</td>
<td>416.67 ns</td>
<td>15 ADC clock cycles</td>
<td>500.0 ns</td>
</tr>
<tr>
<td>10</td>
<td>10.5 ADC clock cycles</td>
<td>350.0 ns</td>
<td>13 ADC clock cycles</td>
<td>433.33 ns</td>
</tr>
<tr>
<td>8</td>
<td>8.5 ADC clock cycles</td>
<td>203.33 ns</td>
<td>11 ADC clock cycles</td>
<td>366.67 ns</td>
</tr>
<tr>
<td>6</td>
<td>6.5 ADC clock cycles</td>
<td>216.67 ns</td>
<td>9 ADC clock cycles</td>
<td>300.0 ns</td>
</tr>
</tbody>
</table>
### 27.4.25 Timing diagrams example (single/continuous modes, hardware/software triggers)

**Figure 250. Single conversions of a sequence, software trigger**

```
<table>
<thead>
<tr>
<th>ADCSTART(1)</th>
<th>EOC</th>
<th>EOS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

ADC state:
- RDY
- CH1
- CH9
- CH10
- CH17
- R DY
- CH1
- CH9
- CH10
- CH17
- RDY

ADC_DR:
- D1
- D9
- D10
- D17

Indicative timings:
- by SW
- by HW

1. EXTEN = 0x0, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

**Figure 251. Continuous conversion of a sequence, software trigger**

```
<table>
<thead>
<tr>
<th>ADCSTART(1)</th>
<th>EOC</th>
<th>EOS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

ADC state:
- READY
- CH1
- CH9
- CH10
- CH17
- CH1
- CH9
- CH10
- STP
- READY
- CH1
- CH9

ADC_DR:
- D1
- D9
- D10
- D17
- D1
- D9
- D1

Indicative timings:
- by SW
- by HW

1. EXTEN = 0x0, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.
1. TRGx (over-frequency) is selected as trigger source, EXTEN = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

1. TRGx is selected as trigger source, EXTEN = 10, CONT = 1
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.
27.4.26 Data management

Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

Data and alignment

At the end of each regular conversion channel (when EOC event occurs), the result of the converted data is stored into the ADC_DR data register which is 16 bits wide.

At the end of each injected conversion channel (when JEOC event occurs), the result of the converted data is stored into the corresponding ADC_JDRy data register which is 16 bits wide.

The ALIGN bit in the ADC_CFGR register selects the alignment of the data stored after conversion. Data can be right- or left-aligned as shown in Figure 254, Figure 255, Figure 256 and Figure 257.

Special case: when left-aligned, the data are aligned on a half-word basis except when the resolution is set to 6-bit. In that case, the data are aligned on a byte basis as shown in Figure 256 and Figure 257.

Note: Left-alignment is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the ALIGN bit value is ignored and the ADC only provides right-aligned data.

Offset

An offset \( y \) \( (y = 1, 2, 3, 4) \) can be applied to a channel by setting the bit OFFSET_EN = 1 into ADC_OFRy register. The channel to which the offset is to be applied is programmed into the bits OFFSET_CH[4:0] of ADC_OFRy register. In this case, the converted value is decreased by the user-defined offset written in the bits OFFSET[11:0]. The result may be a negative value so the read data is signed and the SEXT bit represents the extended sign value.

Note: Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).

Table 235 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

<table>
<thead>
<tr>
<th>Resolution (bits RES[1:0])</th>
<th>Subtraction between raw converted data and offset</th>
<th>Result</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Raw converted Data, left aligned</td>
<td>Offset</td>
<td></td>
</tr>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0]</td>
<td>OFFSET[11:0]</td>
<td>Signed 12-bit data</td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:2],00</td>
<td>OFFSET[11:0]</td>
<td>Signed 10-bit data</td>
</tr>
</tbody>
</table>
When reading data from ADC_DR (regular channel) or from ADC_JDRy (injected channel, y = 1,2,3,4) corresponding to the channel “i”:

- If one of the offsets is enabled (bit OFFSET_EN = 1) for the corresponding channel, the read data is signed.
- If none of the four offsets is enabled for this channel, the read data is not signed.

Figure 254, Figure 255, Figure 256 and Figure 257 show alignments for signed and unsigned data.
**Figure 255. Right alignment (offset enabled, signed value)**

<table>
<thead>
<tr>
<th>12-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
<td>D11</td>
</tr>
<tr>
<td>SEXT</td>
<td>D10</td>
<td>D9</td>
<td>D8</td>
</tr>
<tr>
<td>SEXT</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
</tr>
<tr>
<td>SEXT</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>SEXT</td>
<td>D1</td>
<td>D0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
<td>D9</td>
</tr>
<tr>
<td>SEXT</td>
<td>D8</td>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>SEXT</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
</tr>
<tr>
<td>SEXT</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
<td>D7</td>
</tr>
<tr>
<td>SEXT</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>SEXT</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>SEXT</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
</tr>
<tr>
<td>SEXT</td>
<td>SEXT</td>
<td>SEXT</td>
<td>D5</td>
</tr>
<tr>
<td>SEXT</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>SEXT</td>
<td>D1</td>
<td>D0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 256. Left alignment (offset disabled, unsigned value)**

<table>
<thead>
<tr>
<th>12-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D11</td>
<td>D10</td>
<td>D9</td>
<td>D8</td>
</tr>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>10-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D9</td>
<td>D8</td>
<td>D7</td>
<td>D6</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
</tr>
<tr>
<td>D3</td>
<td>D2</td>
<td>D1</td>
<td>D0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6-bit data</th>
<th>bit15</th>
<th>bit7</th>
<th>bit0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
</tr>
<tr>
<td>D1</td>
<td>D0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Offset compensation

When SATEN bit is set in ADC_OFRy register during offset operation, data are unsigned. All the offset data saturate at 0x000 (in 12-bit mode). When OFFSETPOS bit is set, the offset direction is positive and the data saturate at 0xFFF (in 12-bit mode). In 8-bit mode, data saturate at 0x00 and 0xFF, respectively.

The analog watchdog comparison is performed before the offset compensation.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) notifies when the regular converted data has not been read (by the CPU or the DMA) before ADC_DR FIFO (three stages) is overflowed.

The OVR flag is set when a new conversion completes while ADC_CR register FIFO was full. An interrupt is generated if OVRIE bit is set to 1.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP to 1. Since ADC_DR FIFO features three stages, up to three data are stored in the FIFO.

OVR flag is cleared by software by writing 1 to it.

It is possible to configure if data is preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0: The overrun event preserves the data register from being overwritten: the old data is maintained up to ADC_DR FIFO depth (three stages) and the new conversion is discarded and lost. In this mode, ADC_DR FIFO is disabled. If the FIFO is full, any further conversion is performed but the resulting data is also discarded. EOC
is cleared by reading ADC_DR register. However, the FIFO can still contain previously converted data.

- **OVRMOD = 1**: The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversions is performed normally and the ADC_DR register always contains the latest converted data.

**Figure 258. Example of overrun (OVRMOD = 0)**

<table>
<thead>
<tr>
<th>ADSTART</th>
<th>EOC</th>
<th>OVR</th>
<th>ADSTP</th>
<th>TRGx</th>
<th>ADC state</th>
<th>ADC_DR read access</th>
<th>ADC_DR (FIFO_DATA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RDY</td>
<td>CH1  CH2 CH3 CH4 CH5 CH6 CH7 CH8 STOP RDY</td>
<td>D1  D2  D3  D4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>D5</td>
</tr>
</tbody>
</table>

by s/w  
by h/w  
triggered  

Indicative timings
Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as an error.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and OVR flag should be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

Since converted channel values are stored into a unique data register, it is useful to use DMA for conversion of more than one channel. This avoids the loss of the data already stored in the ADC_DR register.

When the DMA mode is enabled (DMAEN bit set to 1 in the ADC_CFGR register in single ADC mode or MDMA different from 0b00 in dual ADC mode), a DMA request is generated after each conversion of a channel. This allows the transfer of the converted data from the ADC_DR register to the destination location selected by the software.
Despite this, if an overrun occurs (OVR = 1) because the DMA could not serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA. Which means that all the data transferred to the RAM can be considered as valid.

Depending on the configuration of OVRMOD bit, the data is either preserved or overwritten (refer to Section: ADC overrun (OVR, OVRMOD)).

The DMA transfer requests are blocked until the software clears the OVR bit.

Two different DMA modes are proposed depending on the application use and are configured with bit DMACFG of the ADC_CFGR register in single ADC mode, or with bit DMACFG of the ADC_CCR register in dual ADC mode:

- DMA one shot mode (DMACFG = 0)
  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMACFG = 1)
  This mode is suitable when programming the DMA in circular mode.

### DMA one shot mode (DMACFG = 0)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (when a transfer complete interrupt occurs - refer to DMA section) even if a conversion has been started again.

When the DMA transfer is complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- Any ongoing conversion is aborted with partial result discarded.
- No new DMA request is issued to the DMA controller. This avoids generating an overrun error if there are still conversions which are started.
- Scan sequence is stopped and reset.
- The DMA is stopped.

### DMA circular mode (DMACFG = 1)

In this mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register, even if the DMA has reached the last DMA transfer. This allows configuring the DMA in circular mode to handle a continuous analog input data stream.

#### 27.4.27 Managing conversions using the ADF

The ADC conversion results can be transferred directly to the audio digital filter (ADF).

In this case, the ADFCFG bit must be set to 1 and DMAEN bit must be cleared to 0.

The ADC transfers all the 16 bits of the regular data register to the ADF and resets the EOC flag once the transfer is complete.

The data format must be 16-bit signed:

- \( \text{ADC\_DR}[15:12] = \text{sign extended} \)
- \( \text{ADC\_DR}[11] = \text{sign} \)
- \( \text{ADC\_DR}[11:0] = \text{data} \)
To obtain 16-bit signed format in 12-bit ADC mode, the software needs to configure the OFFSET[11:0] to 0x800 after having set OFFSET_EN to 1.

Only right aligned data format is available for the ADF interface (see Figure 255: Right alignment (offset enabled, signed value)).

27.4.28 Dynamic low-power features

Auto-delayed conversion mode (AUTDLY)

The ADC implements an auto-delayed conversion mode controlled by the AUTDLY configuration bit. Auto-delayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there would be risk of encountering an ADC overrun.

When AUTDLY = 1, a new conversion can start only if all the previous data of the same group has been treated:

- For a regular conversion: once the ADC_DR register has been read or if the EOC bit has been cleared (see Figure 260).
- For an injected conversion: when the JEOS bit has been cleared (see Figure 261).

This is a way to automatically adapt the speed of the ADC to the speed of the system which reads the data.

The delay is inserted after each regular conversion (whatever DISCEN = 0 or 1) and after each sequence of injected conversions (whatever JDISCEN = 0 or 1).

Note: There is no delay inserted between each conversions of the injected sequence, except after the last one.

During a conversion, a hardware trigger event (for the same group of conversions) occurring during this delay is ignored.

Note: This is not true for software triggers where it remains possible during this delay to set the bits ADSTART or JADSTART to restart a conversion: it is up to the software to read the data before launching a new conversion.

No delay is inserted between conversions of different groups (a regular conversion followed by an injected conversion or conversely):

- If an injected trigger occurs during the automatic delay of a regular conversion, the injected conversion starts immediately (see Figure 261).
- Once the injected sequence is complete, the ADC waits for the delay (if not ended) of the previous regular conversion before launching a new regular conversion (see Figure 263).

The behavior is slightly different in auto-injected mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see Figure 264).
To stop a conversion in continuous auto-injection mode combined with autodelay mode (JAUTO = 1, CONT = 1 and AUTDLY = 1), follow the following procedure:

1. Wait until JEOS = 1 (no more conversions are restarted)
2. Clear JEOS,
3. Set ADSTP = 1
4. Read the regular data.

If this procedure is not respected, a new regular sequence can restart if JEOS is cleared after ADSTP has been set.

In AUTDLY mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In AUTDLY mode, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

**Figure 260. AUTDLY = 1, regular conversion in continuous mode, software trigger**

1. AUTDLY = 1
2. Regular configuration: EXTEN=0x0 (SW trigger), CONT = 1, CHANNELS = 1,2,3
3. Injected configuration DISABLED
Figure 261. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)

1. AUTDLY = 1
2. Regular configuration: EXTEN=0x1 (HW trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5, 6
Figure 262. AUTODLY = 1, regular HW conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)

1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x1 (HW trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 1, CHANNELS = 5, 6.

1128/3791 RM0477 Rev 6
Figure 263. AUTODLY = 1, regular continuous conversions interrupted by injected conversions

1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3
3. Injected configuration: JEXTEN = 0x1 (HW Trigger), JDISCEN = 0, CHANNELS = 5, 6

Figure 264. AUTODLY = 1 in auto- injected mode (JAUTO = 1)

1. AUTDLY = 1
2. Regular configuration: EXTEN = 0x0 (SW trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6
27.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

**Figure 265. Analog watchdog guarded area**

AWDx flag and interrupt

An interrupt can be enabled for each of the 3 analog watchdogs by setting AWDxIE in the ADC_IER register (x = 1,2,3).

AWDx (x = 1,2,3) flag is cleared by software by writing 1 to it.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

**Description of analog watchdog 1**

The AWD analog watchdog 1 is enabled by setting the AWD1EN bit in the ADC_CFGR register. This watchdog monitors whether either one selected channel or all enabled channels\(^1\) remain within a configured voltage range (window).

**Table 234** shows how the ADC_CFGR registers should be configured to enable the analog watchdog on one or more channels.

<table>
<thead>
<tr>
<th>Channels guarded by the analog watchdog</th>
<th>AWD1SGL bit</th>
<th>AWD1EN bit</th>
<th>JAWD1EN bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All injected channels</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>All regular channels</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>All regular and injected channels</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Single(^1) injected channel</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Single(^1) regular channel</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Single(^1) regular or injected channel</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog status bit is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.
These thresholds are programmed in bits HT1[11:0] and LT1[11:0] of the ADC_TR1 register for the analog watchdog 1. When converting data with a resolution of less than 12 bits (according to bits RES[1:0]), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data (left aligned) before the offset compensation stage. 

Table 235 describes how the comparison is performed for all the possible resolutions for analog watchdog 1.

<table>
<thead>
<tr>
<th>Resolution (bit RES[1:0])</th>
<th>Analog watchdog comparison between:</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>00: 12-bit</td>
<td>DATA[11:0]</td>
<td>LT1[11:0] and HT1[11:0] -</td>
</tr>
<tr>
<td>01: 10-bit</td>
<td>DATA[11:2],00</td>
<td>LT1[11:0] and HT1[11:0] User must configure LT1[1:0] and HT1[1:0] to 00</td>
</tr>
</tbody>
</table>

**Analog watchdog filter for watchdog 1**

When an ADC is configured with only one input channel (selecting several channels in scan mode not allowed), a valid ADC conversion data interval can be configured through the ADC_TR1 register:

- When converted data belong to the interval defined in ADC_TR1, a DMA request is generated.
- Otherwise, no DMA request is issued. RDATA register is updated at each conversion. If data are out-of-range a number of times higher than the value specified in AWDFILT bit of ADC_TR1, the AWDx flag is set and the corresponding interrupt is issued.

**Description of analog watchdog 2 and 3**

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDxCH[18:0] (x = 2,3).

The corresponding watchdog is enabled when any bit of AWDxCH[18:0] (x = 2,3) is set.

They are limited to a resolution of 8 bits and only the 8 MSBs of the thresholds can be programmed into HTx[7:0] and LTx[7:0]. Table 236 describes how the comparison is performed for all the possible resolutions.
ADCy_AWDx_OUT signal output generation

Each analog watchdog is associated to an internal hardware signal ADCy_AWDx_OUT (y = ADC number, x = watchdog number) which is directly connected to the ETR input (external trigger) of some on-chip timers. Refer to the on-chip timers section to understand how to select the ADCy_AWDx_OUT signal as ETR.

ADCy_AWDx_OUT is activated when the associated analog watchdog is enabled:
- ADCy_AWDx_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADCy_AWDx_OUT is reset after the end of the next guarded conversion which is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADCy_AWDx_OUT is also reset when disabling the ADC (when setting ADDIS = 1). Note that stopping regular or injected conversions (setting ADSTP = 1 or JADSTP = 1) has no influence on the generation of ADCy_AWDx_OUT.

Note: AWDx flag is set by hardware and reset by software: AWDx flag has no influence on the generation of ADCy_AWDx_OUT (ex: ADCy_AWDx_OUT can toggle while AWDx flag remains at 1 if the software did not clear the flag).

Figure 266. ADCy_AWDx_OUT signal generation (on all regular channels)
**Figure 267. ADCY_AWDx_OUT signal generation (AWDx flag not cleared by software)**

- Converting regular channels 1,2,3,4,5,6,7
- Regular channels 1,2,3,4,5,6,7 are all guarded

**Figure 268. ADCY_AWDx_OUT signal generation (on a single regular channel)**

- Converting regular channels 1 and 2
- Only channel 1 is guarded

**Figure 269. ADCY_AWDx_OUT signal generation (on all injected channels)**

- Converting the injected channels 1, 2, 3, 4
- All injected channels 1, 2, 3, 4 are guarded
Analog watchdog threshold control

LTx[11:0] and HTx[11:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTx[11:0] and HTx[11:0] are updated during the ADC conversion of the ADC guarded channel, the watchdog function is masked for this conversion. This masking is removed at the next start of conversion, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new interval, no interrupt and AWDx_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion which started after the threshold update. If AWD_xOUT is already asserted, programming the new thresholds does not deassert the AWDx_OUT signal.

Analog watchdog with offset compensation

When the offset compensation is enabled, the analog watchdog compares the threshold before the data compensation.

27.4.30 Oversampler

The oversampling unit performs data pre-processing to offload the CPU. It is able to handle multiple conversions and average them into a single data with increased data width, up to 16-bit.

It provides a result with the following form, where N and M can be adjusted:

\[
\text{Result} = \frac{1}{M} \sum_{n = 0}^{n - N - 1} \text{Conversion}(t_n)
\]

It allows to perform by hardware the following functions: averaging, data rate reduction, SNR improvement, basic filtering.

The oversampling ratio N is defined using the OVFS[2:0] bits in the ADC_CFGR2 register, and can range from 2x to 256x. The division coefficient M consists of a right bit shift up to 8 bits, and is defined using the OVSS[3:0] bits in the ADC_CFGR2 register.

The summation unit can yield a result up to 20 bits (256x 12-bit results), which is first shifted right. It is then truncated to the 16 least significant bits, rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred into the ADC_DR data register.

Note: If the intermediary result after the shifting exceeds 16-bit, the result is truncated as is, without saturation.
Figure 270. 20-bit to 16-bit result truncation

![Diagram showing 20-bit to 16-bit result truncation process]

Figure 271 gives a numerical example of the processing, from a raw 20-bit accumulated data to the final 16-bit result.

Figure 271. Numerical example with 5-bit shift and rounding

![Diagram showing numerical example with 5-bit shift and rounding]

Table 237 gives the data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 237. Maximum output results versus N and M (gray cells indicate truncation)

<table>
<thead>
<tr>
<th>Over sampling ratio</th>
<th>Max Raw data</th>
<th>No-shift OVSS = 0000</th>
<th>1-bit shift OVSS = 0001</th>
<th>2-bit shift OVSS = 0010</th>
<th>3-bit shift OVSS = 0011</th>
<th>4-bit shift OVSS = 0100</th>
<th>5-bit shift OVSS = 0101</th>
<th>6-bit shift OVSS = 0110</th>
<th>7-bit shift OVSS = 0111</th>
<th>8-bit shift OVSS = 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>2x</td>
<td>0x1FFE</td>
<td>0x1FFE</td>
<td>0x0FFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
<td>0x0020</td>
</tr>
<tr>
<td>4x</td>
<td>0x3FFC</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
<td>0x0040</td>
</tr>
<tr>
<td>8x</td>
<td>0x7FF8</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
<td>0x0080</td>
</tr>
<tr>
<td>16x</td>
<td>0xFFF0</td>
<td>0xFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0x0FFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>32x</td>
<td>0x1FFE0</td>
<td>0xFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0xFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>64x</td>
<td>0x3FFC0</td>
<td>0xFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0xFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>128x</td>
<td>0x7FF80</td>
<td>0xFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0xFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
<tr>
<td>256x</td>
<td>0xFFF00</td>
<td>0xFFF0</td>
<td>0x7FF8</td>
<td>0x3FFC</td>
<td>0x1FFE</td>
<td>0xFFF</td>
<td>0x0800</td>
<td>0x0400</td>
<td>0x0200</td>
<td>0x0100</td>
</tr>
</tbody>
</table>

There are no changes for conversion timings in oversampled mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N
Analog-to-digital converters (ADC1/2) RM0477

conversions, with an equivalent delay equal to $N \times T_{\text{CONV}} = N \times (t_{\text{SMPL}} + t_{\text{SAR}})$. The flags are set as follow:

- The end of the sampling phase (EOSMP) is set after each sampling phase
- The end of conversion (EOC) occurs once every $N$ conversions, when the oversampled result is available
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after $N \times$ sequence length conversions total)

**ADC operating modes supported when oversampling (single ADC mode)**

In oversampling mode, most of the ADC operating modes are maintained:

- Single or continuous conversion modes
- ADC conversions start either by software or with triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits in ADC_CFGR register) are accumulated, truncated, rounded and shifted in the same way as 12-bit conversions are

**Note:** The alignment mode is not available when working with oversampled data. The ALIGN bit in ADC_CFGR is ignored and the data are always provided right-aligned.

**Offset correction is not supported in oversampling mode. When ROVSE and/or JOVSE bit is set, the value of the OFFSET_EN bit in ADC_OFRy register is ignored (considered as reset).**

**Analog watchdog**

The analog watchdog functionality is maintained, with the following difference:

- The RES[1:0] bits are ignored, the comparison is always done by using the full 12-bit values HT1[11:0] and LT1[11:0] for AWD1, and the 8-MSB value of HT2/HT3[7:0] and LT2/LT3[7:0] for AWD2 and AWD3.
- The comparison is done on the most significant 12-bit of the 16-bit oversampled results ADC_DR[15:4] for AWD1, and ADC_DR[15:8] for AWD2 and ADW3

**Note:** Care must be taken when using high shifting values, this reduces the comparison range. For instance, if the oversampled result is shifted by 4 bits, thus yielding a 12-bit data right-aligned, the effective analog watchdog comparison can only be performed on 8 bits. The comparison is done between ADC_DR[11:4] and HTx[7:0] / LTx[7:0] (AWD1/2/3), with HT1[11:8] and LT1[11:8] kept reset (AWD1 only).

**Triggered mode**

The averager can also be used for basic filtering purpose. Although not a very powerful filter (slow roll-off and limited stop band attenuation), it can be used as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched mode power supply). For this purpose, a specific discontinuous mode can be enabled with TROVS bit in ADC_CFGR2, to be able to have an oversampling frequency defined by a user and independent from the conversion time itself.

The Figure 272 below shows how conversions are started in response to triggers during discontinuous mode.
If the TROVS bit is set, the content of the DISCEN bit is ignored and considered as 1.

Figure 272. Triggered regular oversampling mode (TROVS bit = 1)

Injected and regular sequencer management when oversampling

In oversampling mode, it is possible to have differentiated behavior for injected and regular sequencers. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

Oversampling regular channels only

The regular oversampling mode bit ROVSM defines how the regular oversampling sequence is resumed if it is interrupted by injected conversion:

- In continued mode, the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that oversampling is complete whatever the injection frequency (providing at least one regular conversion can be completed between triggers);
- In resumed mode, the accumulation restarts from 0 (previous conversions results are ignored). This mode allows to guarantee that all data used for oversampling were converted back-to-back within a single timeslot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be completed and the regular sequencer is blocked.

The Figure 273 gives examples for a 4x oversampling ratio.
**Oversampling Injected channels only**

The Injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.
Oversampling regular and Injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling mode is forced to resumed mode (ROVSM bit ignored), as represented on Figure 274 below.

Figure 274. Regular and injected oversampling modes used simultaneously

![Diagram showing regular and injected oversampling modes used simultaneously.](MS34457V2)

Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected mode oversampling mode must be disabled, and the ROVSM bit is ignored (resumed mode is forced). The JOVSE bit must be reset. The behavior is represented on Figure 275 below.

Figure 275. Triggered regular oversampling with injection

![Diagram showing triggered regular oversampling with injection.](MS34458V4)
Auto-injected mode

It is possible to oversample auto-injected sequences and have all conversions results stored in registers to save a DMA resource. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1, other combinations are not supported. The ROVSM bit is ignored in auto-injected mode. The Figure 276 below shows how the conversions are sequenced.

Figure 276. Oversampling in auto-injected mode

![Diagram showing oversampling in auto-injected mode]

It is possible to have also the triggered mode enabled, using the TROVS bit. In this case, the ADC must be configured as following: JAUTO = 1, DISCEN = 0, JDISCEN = 0, ROVSE = 1, JOVSE = 1 and TROVSE = 1.

Dual ADC modes supported when oversampling

It is possible to have oversampling enabled when working in dual ADC configuration, for the injected simultaneous mode and regular simultaneous mode. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

All other dual ADC modes are not supported when either regular or injected oversampling is enabled (ROVSE = 1 or JOVSE = 1).

Combined modes summary

The Table 238 below summarizes all combinations, including modes not supported.

Table 238. Oversampler operating modes summary

<table>
<thead>
<tr>
<th>Regular Oversampling</th>
<th>Injected Oversampling</th>
<th>Oversampler mode</th>
<th>Triggered Regular mode</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROVSE</td>
<td>JOVSE</td>
<td>ROVSM</td>
<td>TROVS</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = continued</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = resumed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Regular continued mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Triggered regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Not supported</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Injected and regular resumed mode</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Not supported</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Injected oversampling</td>
</tr>
</tbody>
</table>
27.4.31 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see Figure 277).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADCx master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when bits DUAL[4:0] in ADC_CCR register are not equal to zero), the bits CONT, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JQM, JAUTO of the ADC_CFGR register are shared between the master and slave ADC: the bits in the slave ADC are always equal to the corresponding bits of the master ADC.

To start a conversion in dual mode, the user must program the bits EXTEN, EXTSEL, JEXTEN, JEXTSEL of the master ADC only, to configure a software or hardware trigger, and a regular or injected trigger. (the bits EXTEN[1:0] and JEXTEN[1:0] of the slave ADC are don’t care).

In regular simultaneous or interleaved modes: once the user sets bit ADSTART or bit ADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit ADSTART or bit ADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes: once the user sets bit JADSTART or bit JADSTP of the master ADC, the corresponding bit of the slave ADC is also automatically set. However, bit JADSTART or bit JADSTP of the slave ADC is not necessary cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADC_CDR). The status bits can be also read in parallel by reading the dual-mode status register (ADC_CSR).
1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADC_CDR) contains both the master and slave ADC regular converted data.

---

**Figure 277. Dual ADC block diagram**

- **Master ADC**
  - Start trigger mux. (regular group)
  - Internal triggers
  - Internal analog inputs
  - ADCx_INN1, ADCx_INP1
  - ADCx_INN2, ADCx_INP2
  - ADCx_INN16, ADCx_INP16

- **Slave ADC**
  - Start trigger mux. (injected group)
  - Internal triggers
  - Internal analog inputs
  - ADCx_INN1, ADCx_INP1
  - ADCx_INN2, ADCx_INP2

- **Address/data bus**
  - Regular data register (16-bits)
  - Injected data registers (4 x16-bits)

---

*MSv36025V2*
**Injected simultaneous mode**

This mode is selected by programming bits DUAL[4:0] = 00101

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

**Note:** Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master injected sequence is equal to the duration of the slave injected one (like in Figure 278), it is possible for the software to enable only one of the two JEOS interrupt (ex: master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

**Figure 278. Injected simultaneous mode on 4 channels: dual ADC mode**

If JDISCEN = 1, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the
ongoing regular sequence and the associated delay phases are ignored. There is the same behavior for regular sequences occurring on the slave ADC.

Regular simultaneous mode with independent injected

This mode is selected by programming bits DUAL[4:0] = 00110.

This mode is performed on a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on master or on the slave) aborts the current simultaneous conversions, which are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the longer conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Software is notified by interrupts when it can read the data:

- At the end of each conversion event (EOC) on the master ADC, a master EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the master ADC.
- At the end of each conversion event (EOC) on the slave ADC, a slave EOC interrupt is generated (if EOCIE is enabled) and software can read the ADC_DR of the slave ADC.
- If the duration of the master regular sequence is equal to the duration of the slave one (like in Figure 279), it is possible for the software to enable only one of the two EOC interrupt (ex: master EOC) and read both converted data from the Common Data register (ADC_CDR).

It is also possible to read the regular data using the DMA. Two methods are possible:

- Using two DMA channels (one for the master and one for the slave). In this case bits MDMA[1:0] must be kept cleared.
  - Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  - Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.

- Using MDMA mode, which leaves one DMA channel free for other uses:
  - Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
  - A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR)
  - A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CDR register.
  - Both EOC flags are cleared when the DMA reads the ADC_CDR register.

Note: In MDMA mode (MDMA[1:0] = 0b10 or 0b11), the user must program the same number of conversions in the master’s sequence as in the slave’s sequence. Otherwise, the remaining conversions does not generate a DMA request.
If DISCEN = 1 then each “n” simultaneous conversions of the regular sequence require a regular trigger event to occur (“n” is defined by DISCNUM).

This mode can be combined with AUTDLY mode:

- Once a simultaneous conversion of the sequence has ended, the next conversion in the sequence is started only if the common data register, ADC_CDR (or the regular data register of the master ADC) has been read (delay phase).
- Once a simultaneous regular sequence of conversions has ended, a new regular trigger event is accepted only if the common data register (ADC_CDR) has been read (delay phase). Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

It is possible to use the DMA to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multiple-DMA mode is used: bits MDMA must be set to 0b10 or 0b11.

When regular simultaneous mode is combined with AUTDLY mode, it is mandatory for the user to ensure that:

- The number of conversions in the master’s sequence is equal to the number of conversions in the slave’s.
- For each simultaneous conversions of the sequence, the length of the conversion of the slave ADC is inferior to the length of the conversion of the master ADC. Note that the length of the sequence depends on the number of channels to convert and the sampling time and the resolution of each channels.

Note: This combination of regular simultaneous mode and AUTDLY mode is restricted to the use case when only regular channels are programmed: it is forbidden to program injected channels in this combined mode.

**Interleaved mode with independent injected**

This mode is selected by programming bits DUAL[4:0] = 00111.

This mode can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately.
- The slave ADC starts after a delay of several-ADC clock cycles after the sampling phase of the master ADC has complete.

The minimum delay which separates two conversions in interleaved mode is configured in the DELAY bits in the ADC_CCR register. This delay starts counting one half cycle after the end of the sampling phase of the master conversion. This way, an ADC cannot start a
conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

If the CONT bit is set on both master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Note: It is possible to enable only the EOC interrupt of the slave and read the common data register (ADC_CDR). But in this case, the user must ensure that the duration of the conversions are compatible to ensure that inside the sequence, a master conversion is always followed by a slave conversion before a new master conversion restarts. It is recommended to use the MDMA mode.

It is also possible to have the regular data transferred by DMA. In this case, individual DMA requests on each ADC cannot be used and it is mandatory to use the MDMA mode, as following:

- Configure MDMA[1:0] = 0b10 or 0b11 (depending on resolution).
- A single DMA channel is used (the one of the master). Configure the DMA master ADC channel to read the common ADC register (ADC_CDR).
- A single DMA request is generated each time both master and slave EOC events have occurred. At that time, the slave ADC converted data is available in the upper half-word of the ADC_CDR 32-bit register and the master ADC converted data is available in the lower half-word of ADC_CCR register.
- Both EOC flags are cleared when the DMA reads the ADC_CCR register.

**Figure 280. Interleaved mode on one channel in continuous conversion mode: dual ADC mode**
If DISCEN = 1, each "n" simultaneous conversions ("n" is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In this mode, injected conversions are supported. When injection is done (either on master or on slave), both the master and the slave regular conversions are aborted and the sequence is restarted from the master (see Figure 282 below).

**Alternate trigger mode**

This mode is selected by programming bits DUAL[4:0] = 01001.

This mode can be started only on an injected group. The source of external trigger comes from the injected group multiplexer of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN must not be 0x0.
Injected discontinuous mode disabled (JDISCEN = 0 for both ADC)

1. When the 1st trigger occurs, all injected master ADC channels in the group are converted.
2. When the 2nd trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts by converting the injected channels of the master ADC in the group.

Figure 283. Alternate trigger: injected group of each ADC

Note: Regular conversions can be enabled on one or all ADCs. In this case the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between 2 trigger events must be greater than or equal to 1 ADC clock period. The minimum time interval between 2 trigger events that start conversions on the same ADC is the same as in the single ADC mode.
Injected discontinuous mode enabled (JDISCEN = 1 for both ADC)

If the injected discontinuous mode is enabled for both master and slave ADCs:

- When the 1st trigger occurs, the first injected channel of the master ADC is converted.
- When the 2nd trigger occurs, the first injected channel of the slave ADC is converted.
- And so on.

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversions.

If another external trigger occurs after all injected channels in the group have been converted then the alternate trigger process restarts.

Figure 284. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode

Combined regular/injected simultaneous mode

This mode is selected by programming bits DUAL[4:0] = 00001.

It is possible to interrupt the simultaneous conversion of a regular group to start the simultaneous conversion of an injected group.

*Note:* In combined regular/injected simultaneous mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Combined regular simultaneous + alternate trigger mode

This mode is selected by programming bits DUAL[4:0] = 00010.

It is possible to interrupt the simultaneous conversion of a regular group to start the alternate trigger conversion of an injected group. Figure 285 shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion is immediately started after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.
Note: In combined regular simultaneous + alternate trigger mode, one must convert sequences with the same length or ensure that the interval between triggers is longer than the long conversion time of the 2 sequences. Otherwise, the ADC with the shortest sequence may restart while the ADC with the longest sequence is completing the previous conversions.

Figure 285. Alternate + regular simultaneous

If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. Figure 286 shows the behavior in this case (note that the 6th trigger is ignored because the associated alternate conversion is not complete).

Figure 286. Case of trigger occurring during injected conversion

Combined injected simultaneous plus interleaved

This mode is selected by programming bits DUAL[4:0] = 00011

It is possible to interrupt an interleaved conversion with a simultaneous injected event.

In this case the interleaved conversion is interrupted immediately and the simultaneous injected conversion starts. At the end of the injected sequence the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master’s one. Figure 287, Figure 288 and Figure 289 show the behavior using an example.

Caution: In this mode, it is mandatory to use the Common Data Register to read the regular data with a single read access. On the contrary, master-slave data coherency is not guaranteed.
Figure 287. Interleaved single channel CH0 with injected sequence CH11, CH12

Figure 288. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 1: Master interrupted first

Figure 289. Two Interleaved channels (CH1, CH2) with injected sequence CH11, CH12
- case 2: Slave interrupted first
DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to Figure 290: DMA Requests in regular simultaneous mode when MDMA = 0b00).

Figure 290. DMA Requests in regular simultaneous mode when MDMA = 0b00

In simultaneous regular and interleaved modes, it is also possible to save one DMA channel and transfer both data using a single DMA channel. For this MDMA bits must be configured in the ADC_CCR register:

- **MDMA = 0b10**: A single DMA request is generated each time both master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADC_CDR contains the two half-words representing two ADC-converted data items. The slave ADC data take the upper half-word and the master ADC data take the lower half-word.

  This mode is used in interleaved mode and in regular simultaneous mode when resolution is 10-bit or 12-bit.

**Example:**

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: \( \text{ADC}_{-}\text{CDR}[31:0] = \text{SLV}_{-}\text{ADC}_{-}\text{DR}[15:0] \mid \text{MST}_{-}\text{ADC}_{-}\text{DR}[15:0] \)

2nd DMA request: \( \text{ADC}_{-}\text{CDR}[31:0] = \text{SLV}_{-}\text{ADC}_{-}\text{DR}[15:0] \mid \text{MST}_{-}\text{ADC}_{-}\text{DR}[15:0] \)
Figure 291. DMA requests in regular simultaneous mode when MDMA = 0b10

Configuration where each sequence contains only one conversion

Figure 292. DMA requests in interleaved mode when MDMA = 0b10

Configuration where each sequence contains only one conversion
When using MDMA mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion is available.

- **MDMA = 0b11**: This mode is similar to the MDMA = 0b10. The only differences are that on each DMA request (two data items are available), two bytes representing two ADC converted data items are transferred as a half-word.
  
  This mode is used in interleaved and regular simultaneous mode when resolution is 6-bit or when resolution is 8-bit and data is not signed (offsets must be disabled for all the involved channels).

**Example:**

Interleaved dual mode: a DMA request is generated each time 2 data items are available:

1st DMA request: \( \text{ADC}_\text{CDR}[15:0] = \text{SLV}_\text{ADC}_\text{DR}[7:0] | \text{MST}_\text{ADC}_\text{DR}[7:0] \)

2nd DMA request: \( \text{ADC}_\text{CDR}[15:0] = \text{SLV}_\text{ADC}_\text{DR}[7:0] | \text{MST}_\text{ADC}_\text{DR}[7:0] \)

**Overrun detection**

In dual ADC mode (when `DUAL[4:0]` is not equal to b00000), if an overrun is detected on one of the ADCs, the DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the MDMA configuration). It may happen that the EOC bit corresponding to one ADC remains set because the data register of this ADC contains valid data.

**DMA one shot mode/ DMA circular mode when MDMA mode is selected**

When MDMA mode is selected (0b10 or 0b11), bit DMACFG of the ADC_CCR register must also be configured to select between DMA one shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA* (bits DMACFG of master and slave ADC_CFGR are not relevant).

**Stopping the conversions in dual ADC modes**

The user must set the control bits ADSTP/JADSTP of the master ADC to stop the conversions of both ADC in dual ADC mode. The other ADSTP control bit of the slave ADC has no effect in dual ADC mode.

Once both ADC are effectively stopped, the bits ADSTART/JADSTART of the master and slave ADCs are both cleared by hardware.

### 27.4.32 Temperature sensor

The temperature sensor can be used to measure the junction temperature \((T_j)\) of the device.

The temperature sensor is internally connected to the ADC input channels which are used to convert the sensor output voltage to a digital value (see *Table: ADC interconnection* in *Section 27.4.2: ADC pins and internal signals* for more details). When not in use, the sensor can be put in power down mode. It support the temperature range –40 to 125 °C.

*Figure 293* shows the block diagram of connections between the temperature sensor and the ADC.

The temperature sensor output voltage changes linearly with temperature. The offset of this line varies from chip to chip due to process variation (up to 45 °C from one chip to another).
The uncalibrated internal temperature sensor is more suited for applications that detect temperature variations instead of absolute temperatures. To improve the accuracy of the temperature sensor measurement, calibration values are stored in system memory for each device by ST during production.

During the manufacturing process, the calibration data of the temperature sensor and the internal voltage reference are stored in the system memory area. The user application can then read them and use them to improve the accuracy of the temperature sensor or the internal reference (refer to the datasheet for additional information).

The temperature sensor is internally connected to the ADC input channel which is used to convert the sensor’s output voltage to a digital value. Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal temperature sensor.

When not in use, the sensor can be put in power-down mode. *Figure 293* shows the block diagram of the temperature sensor.

**Figure 293. Temperature sensor channel block diagram**

---

**Reading the temperature**

To use the sensor:

1. Select the ADC input channels that is connected to \( V_{\text{SENSE}} \).
2. Program with the appropriate sampling time (refer to electrical characteristics section of the device datasheet).
3. Set the bit in the ADC_CCR register to wake up the temperature sensor from power-down mode.
4. Start the ADC conversion.
5. Read the resulting \( V_{\text{SENSE}} \) data in the ADC data register.
6. Calculate the actual temperature using the following formula:
where:
- TS_CAL2 is the temperature sensor calibration value acquired at TS_CAL2_TEMP.
- TS_CAL1 is the temperature sensor calibration value acquired at TS_CAL1_TEMP.
- TS_DATA is the actual temperature sensor output value converted by ADC.

Refer to the device datasheet for more information about TS_CAL1 and TS_CAL2 calibration points.

Note: The sensor has a startup time after waking from power-down mode before it can output VSENSE at the correct level. The ADC also has a startup time after power-on, so to minimize the delay, the ADEN and bits should be set at the same time.

27.4.33 VBAT supply monitoring

The VBATEN bit in the ADC_CCR register is used to switch to the battery voltage. As the VBAT voltage could be higher than VDDA, to ensure the correct operation of the ADC, the VBAT pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect VBAT/4 to the ADC input channels (see Table: ADC interconnection in Section 27.4.2: ADC pins and internal signals for more details). As a consequence, the converted digital value is one third of the VBAT voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the VBAT/4 voltage.

Figure 294 shows the block diagram of the VBAT sensing feature.
1. The VBATEN bit must be set to enable the conversion of internal channel for VBAT/4.

### 27.4.34 Monitoring the internal voltage reference

It is possible to monitor the internal voltage reference (VREFINT) to have a reference point for evaluating the ADC VREF+ voltage level.

Refer to Table: ADC interconnection in Section 27.4.2: ADC pins and internal signals for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the product datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

*Figure 295* shows the block diagram of the VREFINT sensing feature.

1. The VREFEN bit into ADC_CCR register must be set to enable the conversion of internal channels (VREFINT).
Calculating the actual $V_{REF+}$ voltage using the internal reference voltage

$V_{REF+}$ voltage may be subject to variations or not precisely known. The embedded internal reference voltage $V_{REFINT}$ and its calibration data acquired by the ADC during the manufacturing process at $V_{REF+_charac}$ can be used to evaluate the actual $V_{REF+}$ voltage level.

The following formula gives the actual $V_{REF+}$ voltage supplying the device:

$$V_{REF+} = \frac{V_{REF+_Charac} \times V_{REFINT_CAL}}{V_{REFINT_Data}}$$

Where:
- $V_{REF+_Charac}$ is the value of $V_{REF+}$ voltage characterized at $V_{REFINT}$ during the manufacturing process. It is specified in the device datasheet.
- $V_{REFINT_CAL}$ is the $V_{REFINT}$ calibration value
- $V_{REFINT_Data}$ is the actual $V_{REFINT}$ output value converted by ADC

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between $V_{REF+}$ and the voltage applied on the converted channel.

For applications where $V_{REF+}$ value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from $V_{REF+}$:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL\_SCALE} \times ADC\_DATA$$

By replacing $V_{REF+}$ by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times V_{REFINT_CAL} \times ADC\_DATA}{V_{REFINT_Data} \times FULL\_SCALE}$$

Where:
- $V_{REF+_Charac}$ is the value of $V_{REF+}$ voltage characterized at $V_{REFINT}$ during the manufacturing process.
- $V_{REFINT_CAL}$ is the $V_{REFINT}$ calibration value
- $ADC\_DATA$ is the value measured by the ADC on channel x (right-aligned)
- $V_{REFINT_Data}$ is the actual $V_{REFINT}$ output value converted by the ADC
- $FULL\_SCALE$ is the maximum digital value of the ADC output. For example with 12-bit resolution, it is $2^{12} - 1 = 4095$ or with 8-bit resolution, $2^8 - 1 = 255$.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

27.4.35 Monitoring the supply voltage

ADC2 is connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.
27.5 **ADC in low-power modes**

Table 239. Effect of low-power modes on the ADC

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. DMA requests are functional. ADC interrupts cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The ADC peripheral is not operational. Prior to entering this mode, it is recommended to disable the ADC.</td>
</tr>
<tr>
<td>Standby</td>
<td>The ADC peripheral is powered down and must be reinitialized after exiting from Standby mode.</td>
</tr>
</tbody>
</table>

27.6 **ADC interrupts**

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (flag ADRDY)
- On the end of any conversion for regular groups (flag EOC)
- On the end of a sequence of conversion for regular groups (flag EOS)
- On the end of any conversion for injected groups (flag JEOC)
- On the end of a sequence of conversion for injected groups (flag JEOS)
- When an analog watchdog detection occurs (flag AWD1, AWD2 and AWD3)
- When the end of sampling phase occurs (flag EOSMP)
- When the data overrun occurs (flag OVR)
- When the injected sequence context queue overflows (flag JQOVF)

Separate interrupt enable bits are available for flexibility.
<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop, Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>ADC ready</td>
<td>ADRDY</td>
<td>ADRDYIE</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End of conversion of a regular group</td>
<td>EOC</td>
<td>EOCIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of conversion sequence of a regular group</td>
<td>EOS</td>
<td>EOSIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of conversion of an injected group</td>
<td>JEOC</td>
<td>JEOCIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of conversion sequence of an injected group</td>
<td>JEOS</td>
<td>JEOSIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analog watchdog 1 status bit is set</td>
<td>AWD1</td>
<td>AWD1IE</td>
<td></td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Analog watchdog 2 status bit is set</td>
<td>AWD2</td>
<td>AWD2IE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Analog watchdog 3 status bit is set</td>
<td>AWD3</td>
<td>AWD3IE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of sampling phase</td>
<td>EOSMP</td>
<td>EOSMPIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Injected context queue overflows</td>
<td>JQOVF</td>
<td>JQOVFIE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 27.7 ADC registers (for each ADC)

Refer to Section 1.2 on page 120 for a list of abbreviations used in register descriptions.

### 27.7.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:11</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 10</th>
<th>JQOVF: Injected context queue overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware when an Overflow of the Injected Queue of Context occurs. It is cleared by software writing 1 to it. Refer to Section 27.4.21: Queue of context for injected conversions for more information.</td>
</tr>
<tr>
<td></td>
<td>0: No injected context queue overflow occurred (or the flag event was already acknowledged and cleared by software)</td>
</tr>
<tr>
<td></td>
<td>1: Injected context queue overflow has occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 9</th>
<th>AWD3: Analog watchdog 3 flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT3[7:0] and HT3[7:0] of ADC_TR3 register. It is cleared by software writing 1 to it.</td>
</tr>
<tr>
<td></td>
<td>0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)</td>
</tr>
<tr>
<td></td>
<td>1: Analog watchdog 3 event occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 8</th>
<th>AWD2: Analog watchdog 2 flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT2[7:0] and HT2[7:0] of ADC_TR2 register. It is cleared by software writing 1 to it.</td>
</tr>
<tr>
<td></td>
<td>0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)</td>
</tr>
<tr>
<td></td>
<td>1: Analog watchdog 2 event occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>AWD1: Analog watchdog 1 flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware when the converted voltage crosses the values programmed in the fields LT1[11:0] and HT1[11:0] of ADC_TR1 register. It is cleared by software, writing 1 to it.</td>
</tr>
<tr>
<td></td>
<td>0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)</td>
</tr>
<tr>
<td></td>
<td>1: Analog watchdog 1 event occurred</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>JEOS: Injected channel end of sequence flag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it.</td>
</tr>
<tr>
<td></td>
<td>0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)</td>
</tr>
<tr>
<td></td>
<td>1: Injected conversions complete</td>
</tr>
</tbody>
</table>

---

![ST Logo](RM0477 Rev 6 1161/3791)
Bit 5  **JEOC**: Injected channel end of conversion flag  
This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it or by reading the corresponding ADC_JDRy register.  
0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)  
1: Injected channel conversion complete  

Bit 4  **OVR**: ADC overrun  
This bit is set by hardware when an overrun occurs on a regular channel, meaning that a new conversion has completed while the EOC flag was already set. It is cleared by software writing 1 to it.  
0: No overrun occurred (or the flag event was already acknowledged and cleared by software)  
1: Overrun has occurred  

Bit 3  **EOS**: End of regular sequence flag  
This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it.  
0: Regular Conversions sequence not complete (or the flag event was already acknowledged and cleared by software)  
1: Regular Conversions sequence complete  

Bit 2  **EOC**: End of conversion flag  
This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it or by reading the ADC_DR register.  
0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)  
1: Regular channel conversion complete  

Bit 1  **EOSMP**: End of sampling flag  
This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase.  
0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)  
1: End of sampling phase reached  

Bit 0  **ADRDY**: ADC ready  
This bit is set by hardware after the ADC has been enabled (ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests. It is cleared by software writing 1 to it.  
0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)  
1: ADC is ready to start conversion
# ADC interrupt enable register (ADC_IER)

Address offset: 0x04  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>JQOVFIE</td>
<td>Injected context queue overflow interrupt enable</td>
</tr>
<tr>
<td>29</td>
<td>JEOCIE</td>
<td>End of injected sequence of conversions interrupt enable</td>
</tr>
<tr>
<td>28</td>
<td>EOSMP IE</td>
<td>End of sequence of conversions enabled, must be kept enabled</td>
</tr>
<tr>
<td>27</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>26</td>
<td>OCRIFO IE</td>
<td>Overrun occurred, not interrupted.</td>
</tr>
<tr>
<td>25</td>
<td>OVRIE</td>
<td>Overrun occurred, not interrupted.</td>
</tr>
<tr>
<td>24</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>23</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>22</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>21</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>20</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>19</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>18</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>17</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>16</td>
<td>EOSIE</td>
<td>End of sequence of conversions enabled, not enabled</td>
</tr>
<tr>
<td>15</td>
<td>JEOCIE</td>
<td>End of injected sequence of conversions interrupt enable</td>
</tr>
<tr>
<td>14</td>
<td>AWD3IE</td>
<td>Analog watchdog 3 interrupt enable</td>
</tr>
<tr>
<td>13</td>
<td>AWD2IE</td>
<td>Analog watchdog 2 interrupt enable</td>
</tr>
<tr>
<td>12</td>
<td>AWD1IE</td>
<td>Analog watchdog 1 interrupt enable</td>
</tr>
<tr>
<td>11</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>10</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>9</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>8</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>7</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>6</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>5</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>4</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>3</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>2</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>1</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
<tr>
<td>0</td>
<td>JEOSIE</td>
<td>JEO interrupt enable</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

- **Bit 10 JQOVFIE**: Injected context queue overflow interrupt enable  
  - This bit is set and cleared by software to enable/disable the Injected Context Queue Overflow interrupt.  
  - 0: Injected Context Queue Overflow interrupt disabled  
  - 1: Injected Context Queue Overflow interrupt enabled. An interrupt is generated when the JQOVF bit is set.  
  
  **Note**: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

- **Bit 9 AWD3IE**: Analog watchdog 3 interrupt enable  
  - This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.  
  - 0: Analog watchdog 3 interrupt disabled  
  - 1: Analog watchdog 3 interrupt enabled  
  
  **Note**: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

- **Bit 8 AWD2IE**: Analog watchdog 2 interrupt enable  
  - This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.  
  - 0: Analog watchdog 2 interrupt disabled  
  - 1: Analog watchdog 2 interrupt enabled  
  
  **Note**: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

- **Bit 7 AWD1IE**: Analog watchdog 1 interrupt enable  
  - This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.  
  - 0: Analog watchdog 1 interrupt disabled  
  - 1: Analog watchdog 1 interrupt enabled  
  
  **Note**: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

- **Bit 6 JEOSIE**: End of injected sequence of conversions interrupt enable  
  - This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.  
  - 0: JEOS interrupt disabled  
  - 1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.  
  
  **Note**: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).
Bit 5  **JEOCIE**: End of injected conversion interrupt enable
   This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.
   0: JEOC interrupt disabled.
   1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.
   Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4  **OVRIE**: Overrun interrupt enable
   This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.
   0: Overrun interrupt disabled
   1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.
   Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3  **EOSIE**: End of regular sequence of conversions interrupt enable
   This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.
   0: EOS interrupt disabled
   1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.
   Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2  **EOCIE**: End of regular conversion interrupt enable
   This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.
   0: EOC interrupt disabled
   1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.
   Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1  **EOSMPIE**: End of sampling flag interrupt enable for regular conversions
   This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.
   0: EOSMP interrupt disabled.
   1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.
   Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0  **ADRDYIE**: ADC ready interrupt enable
   This bit is set and cleared by software to enable/disable the ADC Ready interrupt.
   0: ADRDY interrupt disabled
   1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.
   Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
### 27.7.3 ADC control register (ADC_CR)

Address offset: 0x08  
Reset value: 0x2000 0000

|   | ADCAL | ADCALDIF | DEEPPWD | ADVREGEN | Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
|   | ADCAL | ADCALDIF | DEEPPWD | ADVREGEN |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 31|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 30|    | ADCAL | ADCALDIF | DEEPPWD | ADVREGEN |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 29|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 28|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 27|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 26|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 25|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 24|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 23|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 22|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 21|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 20|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 19|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 18|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 17|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 16|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

#### Bit 31 ADCAL: ADC calibration

This bit is set by software to start the calibration of the ADC. Program first the bit ADCALDIF to determine if this calibration applies for single-ended or Differential inputs mode.

It is cleared by hardware after calibration is complete.

- 0: Calibration complete
- 1: Write 1 to calibrate the ADC. Read at 1 means that a calibration is in progress.

**Note:** The software is allowed to launch a calibration by setting ADCAL only when ADEN = 0.

The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1 and ADSTART = 0 and JADSTART = 0 (ADC enabled and no conversion is ongoing).

#### Bit 30 ADCALDIF: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or Differential inputs mode for the calibration.

- 0: Writing ADCAL launches a calibration in single-ended inputs mode.
- 1: Writing ADCAL launches a calibration in Differential inputs mode.

**Note:** The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

#### Bit 29 DEEPPWD: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

- 0: ADC not in Deep-power down
- 1: ADC in Deep-power-down (default reset state)

**Note:** The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).
Bit 28 **ADVREGEN**: ADC voltage regulator enable

This bit is set by software to enable the ADC voltage regulator.

Before performing any operation such as launching a calibration or enabling the ADC, the ADC voltage regulator must first be enabled and the software must wait for the regulator start-up time.

0: ADC Voltage regulator disabled
1: ADC Voltage regulator enabled.

For more details about the ADC voltage regulator enable and disable sequences, refer to Section 27.4.6: ADC Deep-power-down mode (DEEPPWD) and ADC voltage regulator (ADVREGEN).

The software can program this bit field only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 27:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be re-configured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

0: No ADC stop injected conversion command ongoing
1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

**Note:** The software is allowed to set JADSTP only when JADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting an injected conversion and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP Command).

It is cleared by hardware when the conversion is effectively discarded and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

0: No ADC stop regular conversion command ongoing
1: Write 1 to stop regular conversions ongoing. Read 1 means that an ADSTP command is in progress.

**Note:** The software is allowed to set ADSTP only when ADSTART = 1 and ADDIS = 0 (ADC is enabled and eventually converting a regular conversion and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP)
Bit 3  **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN, a conversion immediately starts (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (JEXTSEL = 0x0): at the assertion of the End of Injected Conversion Sequence (JEOS) flag.
- in all cases: after the execution of the JADSTP command, at the same time that JADSTP is cleared by hardware.

0: No ADC injected conversion is ongoing.
1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

**Note:** The software is allowed to set JADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared).
Bit 2  **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN, a conversion immediately starts (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

It is cleared by hardware:

- in single conversion mode when software trigger is selected (EXTSEL = 0x0): at the assertion of the End of Regular Conversion Sequence (EOS) flag.
- in all cases: after the execution of the ADSTP command, at the same time that ADSTP is cleared by hardware.

0: No ADC regular conversion is ongoing.
1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

*Note:* The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC)

In auto-injection mode (JAUTO = 1), regular and auto-injected conversions are started by setting bit ADSTART (JADSTART must be kept cleared)

Bit 1  **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and put it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing
1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

*Note:* The software is allowed to set ADDIS only when ADEN = 1 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)

Bit 0  **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the flag ADRDY has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)
1: Write 1 to enable the ADC.

*Note:* The software is allowed to set ADEN only when all bits of ADC_CR registers are 0 (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0) except for bit ADVREGEN which must be 1 (and the software must have wait for the startup time of the voltage regulator)
### 27.7.4 ADC configuration register (ADC_CFGR)

Address offset: 0x0C
Reset value: 0x8000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>JQDIS: Injected queue disable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software to disable the injected queue mechanism:</td>
</tr>
<tr>
<td></td>
<td>0: Injected queue enabled</td>
</tr>
<tr>
<td></td>
<td>1: Injected queue disabled</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing). A set or reset of JQDIS bit causes the injected queue to be flushed and the JSQR register is cleared.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 30:26</th>
<th>AWD1CH[4:0]: Analog watchdog 1 channel selection</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.</td>
</tr>
<tr>
<td></td>
<td>00000: ADC analog input channel 0 monitored by AWD1 (available on ADC1 only)</td>
</tr>
<tr>
<td></td>
<td>00001: ADC analog input channel 1 monitored by AWD1</td>
</tr>
<tr>
<td></td>
<td>......</td>
</tr>
<tr>
<td></td>
<td>10010: ADC analog input channel 18 monitored by AWD1</td>
</tr>
<tr>
<td></td>
<td>others: reserved, must not be used</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value. The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers. The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 25</th>
<th>JAUTO: Automatic injected group conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.</td>
</tr>
<tr>
<td></td>
<td>0: Automatic injected group conversion disabled</td>
</tr>
<tr>
<td></td>
<td>1: Automatic injected group conversion enabled</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no regular nor injected conversion is ongoing). When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit JAUTO of the slave ADC is no more writable and its content is equal to the bit JAUTO of the master ADC.</td>
</tr>
</tbody>
</table>
Bit 24  **JAWD1EN**: Analog watchdog 1 enable on injected channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on injected channels
1: Analog watchdog 1 enabled on injected channels

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

Bit 23  **AWD1EN**: Analog watchdog 1 enable on regular channels
This bit is set and cleared by software
0: Analog watchdog 1 disabled on regular channels
1: Analog watchdog 1 enabled on regular channels

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).*

Bit 22  **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels
This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels
0: Analog watchdog 1 enabled on all channels
1: Analog watchdog 1 enabled on a single channel

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 21  **JQM**: JSQR queue mode
This bit is set and cleared by software.
It defines how an empty Queue is managed.
0: JSQR mode 0: The Queue is never empty and maintains the last written configuration into JSQR.
1: JSQR mode 1: The Queue can be empty and when this occurs, the software and hardware triggers of the injected sequence are both internally disabled just after the completion of the last valid injected sequence.

Refer to *Section 27.4.21: Queue of context for injected conversions* for more information.

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit JQM of the slave ADC is no more writable and its content is equal to the bit JQM of the master ADC.

Bit 20  **JDISCEN**: Discontinuous mode on injected channels
This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.
0: Discontinuous mode on injected channels disabled
1: Discontinuous mode on injected channels enabled

*Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).*

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

When dual mode is enabled (bits DUAL of ADC_CCR register are not equal to zero), the bit JDISCEN of the slave ADC is no more writable and its content is equal to the bit JDISCEN of the master ADC.
Bits 19:17 **DISCNUM[2:0]**: Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

- 000: 1 channel
- 001: 2 channels
- ...
- 111: 8 channels

**Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bits DISCNUM[2:0] of the slave ADC are no more writable and their content is equal to the bits DISCNUM[2:0] of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

- 0: Discontinuous mode for regular channels disabled
- 1: Discontinuous mode for regular channels enabled

**Note:** It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.

It is not possible to use both auto-injected mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit DISCEN of the slave ADC is no more writable and its content is equal to the bit DISCEN of the master ADC.

Bit 15 **ALIGN**: Data alignment

This bit is set and cleared by software to select right or left alignment. Refer to Section: Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN).

- 0: Right alignment
- 1: Left alignment

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the Auto Delayed Conversion mode.

- 0: Auto-delayed conversion mode off
- 1: Auto-delayed conversion mode on

**Note:** The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit AUTDLY of the slave ADC is no more writable and its content is equal to the bit AUTDLY of the master ADC.
Bit 13 CONT: Single / continuous conversion mode for regular conversions
This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.
0: Single conversion mode
1: Continuous conversion mode
Note: It is not possible to have both discontinuous mode and continuous mode enabled: it is forbidden to set both DISCEN = 1 and CONT = 1.
The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).
When dual mode is enabled (DUAL bits in ADC_CCR register are not equal to zero), the bit CONT of the slave ADC is no more writable and its content is equal to the bit CONT of the master ADC.

Bit 12 OVRMOD: Overrun mode
This bit is set and cleared by software and configure the way data overrun is managed.
0: ADC_DR register is preserved with the old data when an overrun is detected.
1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.
Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 EXTEN[1:0]: External trigger enable and polarity selection for regular channels
These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.
00: Hardware trigger detection disabled (conversions can be launched by software)
01: Hardware trigger detection on the rising edge
10: Hardware trigger detection on the falling edge
11: Hardware trigger detection on both the rising and falling edges
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 EXTSEL[4:0]: External trigger selection for regular group
These bits select the external event used to trigger the start of conversion of a regular group:
00000: adc_ext_trg0
00001: adc_ext_trg1
00010: adc_ext_trg2
00011: adc_ext_trg3
00100: adc_ext_trg4
00101: adc_ext_trg5
00110: adc_ext_trg6
00111: adc_ext_trg7
...
11111: adc_ext_trg31
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 4:3 RES[1:0]: Data resolution
These bits are written by software to select the resolution of the conversion.
00: 12-bit
01: 10-bit
10: 8-bit
11: 6-bit
Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Bit 2  **ADFCFG**: ADF mode configuration
This bit is set and cleared by software to enable the ADF mode. It is effective only when DMAEN = 0.
0: ADF mode disabled
1: ADF mode enabled
Note: To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0.

Bit 1  **DMACFG**: Direct memory access configuration
This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.
0: DMA One Shot mode selected
1: DMA Circular mode selected
For more details, refer to **Section : Managing conversions using the DMA**
Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
In dual-ADC modes, this bit is not relevant and replaced by control bit DMACFG of the ADC_CCR register.

Bit 0  **DMAEN**: Direct memory access enable
This bit is set and cleared by software to enable the generation of DMA requests. This allows to use the DMA to manage automatically the converted data. For more details, refer to **Section : Managing conversions using the DMA**.
0: DMA disabled
1: DMA enabled
Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
In dual-ADC modes, this bit is not relevant and replaced by control bits MDMA[1:0] of the ADC_CCR register.

### 27.7.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **SMPTRG**: Sampling time control trigger mode
   This bit is set and cleared by software to enable the sampling time control trigger mode.
   0: Sampling time control trigger mode disabled
   1: Sampling time control trigger mode enabled
   The sampling time starts on the trigger rising edge, and the conversion on the trigger falling edge.
   EXTEN bit should be set to 01. BULB bit must not be set when the SMPTRG bit is set.
   When EXTEN bit is set to 00, set SWTRIG to start the sampling and clear SWTRIG bit to start the conversion.
   *Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 26 **BULB**: Bulb sampling mode
   This bit is set and cleared by software to enable the bulb sampling mode.
   0: Bulb sampling mode disabled
   1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.
   SAMPTRG bit must not be set when the BULB bit is set.
   The very first ADC conversion is performed with the sampling time specified in SMPx bits.
   *Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 25 **SWTRIG**: Software trigger bit for sampling time control trigger mode
   This bit is set and cleared by software to enable the bulb sampling mode.
   0: Software trigger starts the conversion for sampling time control trigger mode
   1: Software trigger starts the sampling for sampling time control trigger mode
   *Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 24:17 Reserved, must be kept at reset value.

Bits 16:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode
   This bit is set and cleared by software to select the regular oversampling mode.
   0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)
   1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)
   *Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 9 **TROVS**: Triggered Regular oversampling
   This bit is set and cleared by software to enable triggered oversampling
   0: All oversampled conversions for a channel are done consecutively following a trigger
   1: Each oversampled conversion for a channel needs a new trigger
   *Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*
Bits 8:5 **OVSS[3:0]:** Oversampling shift
This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.
- 0000: No shift
- 0001: Shift 1-bit
- 0010: Shift 2-bits
- 0011: Shift 3-bits
- 0100: Shift 4-bits
- 0101: Shift 5-bits
- 0110: Shift 6-bits
- 0111: Shift 7-bits
- 1000: Shift 8-bits
Other codes reserved
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 4:2 **OVSR[2:0]:** Oversampling ratio
This bitfield is set and cleared by software to define the oversampling ratio.
- 000: 2x
- 001: 4x
- 010: 8x
- 011: 16x
- 100: 32x
- 101: 64x
- 110: 128x
- 111: 256x
*Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bit 1 **JOVSE:** Injected oversampling Enable
This bit is set and cleared by software to enable injected oversampling.
- 0: Injected oversampling disabled
- 1: Injected oversampling enabled
*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*

Bit 0 **ROVSE:** Regular oversampling Enable
This bit is set and cleared by software to enable regular oversampling.
- 0: Regular oversampling disabled
- 1: Regular oversampling enabled
*Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing)*
27.7.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31 (SMPPLUS)</th>
<th>Addition of one clock cycle to the sampling time.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 2.5 ADC clock cycle sampling time becomes 3.5 ADC clock cycles for the ADC_SMPR1 and ADC_SMPR2 registers.</td>
<td></td>
</tr>
<tr>
<td>0: The sampling time remains set to 2.5 ADC clock cycles remains</td>
<td></td>
</tr>
<tr>
<td>To make sure no conversion is ongoing, the software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0.</td>
<td></td>
</tr>
</tbody>
</table>

| Bit 30 Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 29:0 SMPx[2:0]: Channel x sampling time selection (x = 9 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits are written by software to select the sampling time individually for each channel.</td>
</tr>
<tr>
<td>During sample cycles, the channel selection bits must remain unchanged.</td>
</tr>
<tr>
<td>000: 2.5 ADC clock cycles</td>
</tr>
<tr>
<td>001: 6.5 ADC clock cycles</td>
</tr>
<tr>
<td>010: 12.5 ADC clock cycles</td>
</tr>
<tr>
<td>011: 24.5 ADC clock cycles</td>
</tr>
<tr>
<td>100: 47.5 ADC clock cycles</td>
</tr>
<tr>
<td>101: 92.5 ADC clock cycles</td>
</tr>
<tr>
<td>110: 247.5 ADC clock cycles</td>
</tr>
<tr>
<td>111: 640.5 ADC clock cycles</td>
</tr>
<tr>
<td>Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
<tr>
<td>Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.</td>
</tr>
</tbody>
</table>

27.7.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x18
Reset value: 0x0000 0000

| Bit 31 Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 29:0 SMPx[2:0]: Channel x sampling time selection (x = 9 to 0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>These bits are written by software to select the sampling time individually for each channel.</td>
</tr>
<tr>
<td>During sample cycles, the channel selection bits must remain unchanged.</td>
</tr>
<tr>
<td>000: 2.5 ADC clock cycles</td>
</tr>
<tr>
<td>001: 6.5 ADC clock cycles</td>
</tr>
<tr>
<td>010: 12.5 ADC clock cycles</td>
</tr>
<tr>
<td>011: 24.5 ADC clock cycles</td>
</tr>
<tr>
<td>100: 47.5 ADC clock cycles</td>
</tr>
<tr>
<td>101: 92.5 ADC clock cycles</td>
</tr>
<tr>
<td>110: 247.5 ADC clock cycles</td>
</tr>
<tr>
<td>111: 640.5 ADC clock cycles</td>
</tr>
<tr>
<td>Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).</td>
</tr>
<tr>
<td>Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.</td>
</tr>
</tbody>
</table>
27.7.8 ADC watchdog threshold register 1 (ADC_TR1)

Address offset: 0x20

Reset value: 0x0FFF 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 26:0 **SMPx[2:0]**: Channel x sampling time selection (x = 18 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

- 000: 2.5 ADC clock cycles
- 001: 6.5 ADC clock cycles
- 010: 12.5 ADC clock cycles
- 011: 24.5 ADC clock cycles
- 100: 47.5 ADC clock cycles
- 101: 92.5 ADC clock cycles
- 110: 247.5 ADC clock cycles
- 111: 640.5 ADC clock cycles

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing). Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.*

Bits 27:16 **HT1[11:0]**: Analog watchdog 1 higher threshold

These bits are written by software to define the higher threshold for the analog watchdog 1.

Refer to Section 27.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*
Bit 15  Reserved, must be kept at reset value.

Bits 14:12  **AWDFILT[2:0]**: Analog watchdog filtering parameter
            This bit is set and cleared by software.
            000: No filtering
            001: two consecutive detection generates an AWDx flag or an interrupt
            ...
            111: Eight consecutive detection generates an AWDx flag or an interrupt

*Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 11:0  **LT1[11:0]**: Analog watchdog 1 lower threshold
            These bits are written by software to define the lower threshold for the analog watchdog 1.
            Refer to Section 27.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

### 27.7.9  **ADC watchdog threshold register 2 (ADC_TR2)**

Address offset: 0x24
Reset value: 0x00FF 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**HT2[7:0]**

**LT2[7:0]**

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **HT2[7:0]**: Analog watchdog 2 higher threshold
            These bits are written by software to define the higher threshold for the analog watchdog 2.
            Refer to Section 27.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **LT2[7:0]**: Analog watchdog 2 lower threshold
            These bits are written by software to define the lower threshold for the analog watchdog 2.
            Refer to Section 27.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

*Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).*
27.7.10 ADC watchdog threshold register 3 (ADC_TR3)

Address offset: 0x28
Reset value: 0x00FF 0000

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **HT3[7:0]**: Analog watchdog 3 higher threshold
These bits are written by software to define the higher threshold for the analog watchdog 3.
Refer to Section 27.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWD2CH, AWD3CH, AWD_HTx, AWD_LTx, AWDx)

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **LT3[7:0]**: Analog watchdog 3 lower threshold
These bits are written by software to define the lower threshold for the analog watchdog 3.
This watchdog compares the 8-bit of LT3 with the 8 MSB of the converted data.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

27.7.11 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x30
Reset value: 0x0000 0000

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]**: 4th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.
Bits 22:18 **SQ3[4:0]**: 3rd conversion in regular sequence
- These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the regular conversion sequence.
- Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]**: 2nd conversion in regular sequence
- These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the regular conversion sequence.
- Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]**: 1st conversion in regular sequence
- These bits are written by software with the channel number (0 to 18) assigned as the 1st in the regular conversion sequence.
- Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **L[3:0]**: Regular channel sequence length
- These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
- 0000: 1 conversion
- 0001: 2 conversions
- ...
- 1111: 16 conversions
- Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

**Note:** Some channels are not connected physically and must not be selected for conversion.

### 27.7.12 ADC regular sequence register 2 (ADC_SQR2)

**Address offset:** 0x34

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Note:** The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).
Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]**: 9th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 9th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]**: 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 8th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]**: 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 7th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]**: 6th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 6th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]**: 5th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 5th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.

### 27.7.13 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x38

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

ST
Note: Some channels are not connected physically and must not be selected for conversion.

### 27.7.14 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x3C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SQ16[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).
Bits 31:11  Reserved, must be kept at reset value.

Bits 10:6  **SQ16[4:0]:** 16th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 16th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5  Reserved, must be kept at reset value.

Bits 4:0  **SQ15[4:0]:** 15th conversion in regular sequence
These bits are written by software with the channel number (0 to 18) assigned as the 15th in the regular conversion sequence.

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

*Note:* Some channels are not connected physically and must not be selected for conversion.

### 27.7.15  ADC regular data register (ADC_DR)

Address offset: 0x40  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**RDATA[15:0]**

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **RDATA[15:0]:** Regular data converted
These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 27.4.26: Data management.*

### 27.7.16  ADC injected sequence register (ADC_JSQR)

Address offset: 0x4C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**JSQ2[0]:** Res.  
**JSQ1[4:0]:**  
**JEXTEN[1:0]:**  
**JEXTSEL[4:0]:**  
**JL[1:0]:**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:27  **JSQ4[4:0]**: 4th conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 4th in the injected conversion sequence.
*Note:*  The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26  Reserved, must be kept at reset value.

Bits 25:21  **JSQ3[4:0]**: 3rd conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 3rd in the injected conversion sequence.
*Note:*  The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20  Reserved, must be kept at reset value.

Bits 19:15  **JSQ2[4:0]**: 2nd conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 2nd in the injected conversion sequence.
*Note:*  The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14  Reserved, must be kept at reset value.

Bits 13:9  **JSQ1[4:0]**: 1st conversion in the injected sequence
These bits are written by software with the channel number (0 to 18) assigned as the 1st in the injected conversion sequence.
*Note:*  The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).
Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: If JQDIS = 0 (queue enabled), hardware and software trigger detection disabled. Otherwise, the queue is disabled as well as hardware trigger detection (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

**Note:** The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

If JQM = 1 and if the Queue of Context becomes empty, the software and hardware triggers of the injected sequence are both internally disabled (refer to Section 27.4.21: Queue of context for injected conversions)

Bits 6:2 **JEXTSEL[4:0]**: External Trigger Selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

... 

11111: adc_jext_trg31

**Note:** The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JL[1:0]**: Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

**Note:** The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

**Note:** Some channels are not connected physically and must not be selected for conversion.
27.7.17 ADC offset y register (ADC_OFRy)

Address offset: 0x60 + 0x04 * (y -1), (y = 1 to 4)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFFSET_EN</td>
<td>OFFSET_CH[4:0]</td>
<td>SATEN</td>
<td>OFFSE TPOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 OFFSET_EN: Offset y enable
This bit is written by software to enable or disable the offset programmed into bits OFFSET[11:0].

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 30:26 OFFSET_CH[4:0]: Channel selection for the data offset y
These bits are written by software to define the channel to which the offset programmed into bits OFFSET[11:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Some channels are not connected physically and must not be selected for the data offset y.
If OFFSET_EN is set, it is not allowed to select the same channel for different ADC_OFRy registers.

Bit 25 SATEN: Saturation enable
This bit is set and cleared by software to enable the saturation at 0x000 and 0xFFF for the offset function.
0: No saturation control, offset result can be signed
1: Saturation enabled, offset result unsigned and saturated at 0x000 and 0xFFF

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).
Bit 24 **OFFSETPOS**: Positive offset

This bit is set and cleared by software to enable the positive offset.

0: Negative offset
1: Positive offset

**Note:** The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:0 **OFFSET[11:0]**: Data offset y for the channel programmed into bits OFFSET_CH[4:0]

These bits are written by software to define the offset to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset must be programmed in the bits OFFSET_CH[4:0]. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

**Note:** The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

If several offset (OFFSET) point to the same channel, only the offset with the lowest x value is considered for the subtraction.

Ex: if OFFSET1_CH[4:0] = 4 and OFFSET2_CH[4:0] = 4, this is OFFSET1[11:0] which is subtracted when converting channel 4.

### 27.7.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x80 + 0x04 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>JDATA[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **JDATA[15:0]**: Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left -or right-aligned as described in Section 27.4.26: Data management.
### 27.7.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0xA0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**AWD2CH[15:0]**  

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**Bits 31:19** Reserved, must be kept at reset value.  
**Bits 18:0** **AWD2CH[18:0]:** Analog watchdog 2 channel selection  
These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.  
AWD2CH[i] = 0: ADC analog input channel i is not monitored by AWD2  
AWD2CH[i] = 1: ADC analog input channel i is monitored by AWD2  
When AWD2CH[18:0] = 000..0, the analog Watchdog 2 is disabled  
Note: The channels selected by AWD2CH must be also selected into the SQRI or JSQRi registers.  
The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).  
Some channels are not connected physically and must not be selected for the analog watchdog.

### 27.7.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0xA4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**AWD3CH[15:0]**  

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
27.7.21 ADC Differential mode selection register (ADC_DIFSEL)

Address offset: 0xB0

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:0 DIFSEL[18:0]: Differential mode for channels 18 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or Differential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode
DIFSEL[i] = 1: ADC analog input channel i is configured in Differential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).
The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).
27.7.22 ADC calibration factors (ADC_CALFACT)

Address offset: 0xB4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0xB4</th>
<th>Reset value: 0x0000 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALFACT_D[6:0]</td>
<td>rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>CALFACT_S[6:0]</td>
<td>rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

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

Bits 22:16 CALFACT_D[6:0]: Calibration Factors in differential mode
These bits are written by hardware or by software.
Once a differential inputs calibration is complete, they are updated by hardware with the calibration factors.
Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new differential calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

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

Bits 6:0 CALFACT_S[6:0]: Calibration Factors In single-ended mode
These bits are written by hardware or by software.
Once a single-ended inputs calibration is complete, they are updated by hardware with the calibration factors.
Software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored into the analog ADC, it is then applied once a new single-ended calibration is launched.

Note: The software is allowed to write these bits only when ADEN = 1, ADSTART = 0 and JADSTART = 0 (ADC is enabled and no calibration is ongoing and no conversion is ongoing).

27.7.23 ADC option register (ADC_OR)

Address offset: 0xC8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset: 0xC8</th>
<th>Reset value: 0x0000 0000</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 OP0: Output Polarity
These are written by hardware or by software.
These bits are used for selecting the output polarity of the ADC. The default value is 0, which means that the output is high when the analog input is high.

Note: The software is allowed to write these bits only when ADEN = 1 and ADSTART = 0 (ADC is enabled and no conversion is ongoing).
27.8 **ADC common registers**

These registers define the control and status registers common to master and slave ADCs:

### 27.8.1 ADC common status register (ADC_CSR)

Address offset: 0x300

Reset value: 0x0000 0000

This register provides an image of the status bits of the different ADC. Nevertheless it is read-only and does not allow to clear the different status bits. Instead each status bit must be cleared by writing 0 to it in the corresponding ADC_ISR register.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td><strong>JQOVF_SLV</strong>: Injected Context Queue Overflow flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.</td>
</tr>
<tr>
<td>25</td>
<td><strong>AWD3_SLV</strong>: Analog watchdog 3 flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.</td>
</tr>
<tr>
<td>24</td>
<td><strong>AWD2_SLV</strong>: Analog watchdog 2 flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.</td>
</tr>
<tr>
<td>23</td>
<td><strong>AWD1_SLV</strong>: Analog watchdog 1 flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.</td>
</tr>
<tr>
<td>22</td>
<td><strong>JEOS_SLV</strong>: End of injected sequence flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.</td>
</tr>
<tr>
<td>21</td>
<td><strong>JEOC_SLV</strong>: End of injected conversion flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.</td>
</tr>
<tr>
<td>20</td>
<td><strong>OVR_SLV</strong>: Overrun flag of the slave ADC</td>
</tr>
<tr>
<td></td>
<td>This bit is a copy of the OVR bit in the corresponding ADC_ISR register.</td>
</tr>
</tbody>
</table>
Bit 19  **EOS_SLV**: End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18  **EOC_SLV**: End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17  **EOSMP_SLV**: End of Sampling phase flag of the slave ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 16  **ADRDY_SLV**: Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

Bits 15:11  Reserved, must be kept at reset value.

Bit 10  **JQOVF_MST**: Injected Context Queue Overflow flag of the master ADC
This bit is a copy of the JQOVF bit in the corresponding ADC_ISR register.

Bit 9  **AWD3_MST**: Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8  **AWD2_MST**: Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7  **AWD1_MST**: Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6  **JEOS_MST**: End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5  **JEOC_MST**: End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4  **OVR_MST**: Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3  **EOS_MST**: End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2  **EOC_MST**: End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1  **EOSMP_MST**: End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0  **ADRDY_MST**: Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.
27.8.2 ADC common control register (ADC_CCR)

Address offset: 0x308
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable
This bit is set and cleared by software to control.
0: VBAT channel disabled
1: VBAT channel enabled

Bit 23 **TSEN**: VSENSE enable
This bit is set and cleared by software to control VSENSE.
0: Temperature sensor channel disabled
1: Temperature sensor channel enabled

Bit 22 **VREFEN**: VREFINT enable
This bit is set and cleared by software to enable/disable the VREFINT channel.
0: VREFINT channel disabled
1: VREFINT channel enabled

Bits 21:18 **PRESC[3:0]**: ADC prescaler
These bits are set and cleared by software to select the frequency of the clock to the ADC.
The clock is common for all the ADCs.
0000: input ADC clock not divided
0001: input ADC clock divided by 2
0010: input ADC clock divided by 4
0011: input ADC clock divided by 6
0100: input ADC clock divided by 8
0101: input ADC clock divided by 10
0110: input ADC clock divided by 12
0111: input ADC clock divided by 16
1000: input ADC clock divided by 32
1001: input ADC clock divided by 64
1010: input ADC clock divided by 128
1011: input ADC clock divided by 256
other: reserved

Note: The software is allowed to write these bits only when the ADC is disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0). The ADC prescaler value is applied only when CKMODE[1:0] = 0b00.
Bits 17:16 **CKMODE[1:0]**: ADC clock mode

These bits are set and cleared by software to define the ADC clock scheme (which is common to both master and slave ADCs):

- **00**: adc_ker_ck (x = 1/2) (Asynchronous clock mode), generated at product level (refer to Section 6: Reset and clock control (RCC))
- **01**: adc_hclk/1 (Synchronous clock mode). This configuration must be enabled only if the AHB clock prescaler is set to 1 (HPRE[3:0] = 0XXX in RCC_CFGR register) and if the system clock has a 50% duty cycle.
- **10**: adc_hclk/2 (Synchronous clock mode)
- **11**: adc_hclk/4 (Synchronous clock mode)

In all synchronous clock modes, there is no jitter in the delay from a timer trigger to the start of a conversion.

*Note:* The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 15:14 **MDMA[1:0]**: Direct memory access mode for dual ADC mode

This bitfield is set and cleared by software. Refer to the DMA controller section for more details.

- **00**: MDMA mode disabled
- **01**: Reserved
- **10**: MDMA mode enabled for 12 and 10-bit resolution
- **11**: MDMA mode enabled for 8 and 6-bit resolution

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 13 **DMACFG**: DMA configuration (for dual ADC mode)

This bit is set and cleared by software to select between two DMA modes of operation and is effective only when DMAEN = 1.

- **0**: DMA One Shot mode selected
- **1**: DMA Circular mode selected

For more details, refer to Section : Managing conversions using the DMA

*Note:* The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 12 Reserved, must be kept at reset value.
Bits 11:8 **DELAY[3:0]**: Delay between 2 sampling phases

These bits are set and cleared by software. These bits are used in dual interleaved modes.
Refer to **Table 241** for the value of ADC resolution versus DELAY bits values.

*Note:* The software is allowed to write these bits only when the ADCs are disabled
(ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]**: Dual ADC mode selection

These bits are written by software to select the operating mode. 00000 corresponds to
independent mode. Values 00001 to 01001 correspond to dual mode, master and slave
ADCs working together.
00000: Independent mode
00001: Combined regular simultaneous + injected simultaneous mode
00010: Combined regular simultaneous + alternate trigger mode
00011: Combined interleaved mode + injected simultaneous mode
00100: Reserved
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
Others: Reserved, must not be used

*Note:* The software is allowed to write these bits only when the ADCs are disabled
(ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

**Table 241. DELAY bits versus ADC resolution**

<table>
<thead>
<tr>
<th>DELAY bits</th>
<th>12-bit resolution</th>
<th>10-bit resolution</th>
<th>8-bit resolution</th>
<th>6-bit resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1 * Tadc_ker_ck</td>
<td>1 * Tadc_ker_ck</td>
<td>1 * Tadc_ker_ck</td>
<td>1 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0001</td>
<td>2 * Tadc_ker_ck</td>
<td>2 * Tadc_ker_ck</td>
<td>2 * Tadc_ker_ck</td>
<td>2 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0010</td>
<td>3 * Tadc_ker_ck</td>
<td>3 * Tadc_ker_ck</td>
<td>3 * Tadc_ker_ck</td>
<td>3 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0011</td>
<td>4 * Tadc_ker_ck</td>
<td>4 * Tadc_ker_ck</td>
<td>4 * Tadc_ker_ck</td>
<td>4 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0100</td>
<td>5 * Tadc_ker_ck</td>
<td>5 * Tadc_ker_ck</td>
<td>5 * Tadc_ker_ck</td>
<td>5 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0101</td>
<td>6 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0110</td>
<td>7 * Tadc_ker_ck</td>
<td>7 * Tadc_ker_ck</td>
<td>7 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>0111</td>
<td>8 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1000</td>
<td>9 * Tadc_ker_ck</td>
<td>9 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1001</td>
<td>10 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1010</td>
<td>11 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>1011</td>
<td>12 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
<tr>
<td>others</td>
<td>12 * Tadc_ker_ck</td>
<td>10 * Tadc_ker_ck</td>
<td>8 * Tadc_ker_ck</td>
<td>6 * Tadc_ker_ck</td>
</tr>
</tbody>
</table>
27.8.3 ADC common regular data register for dual mode (ADC_CDR)

Address offset: 0x30C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RDATA_SLV[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 **RDATA_SLV[15:0]**: Regular data of the slave ADC
In dual mode, these bits contain the regular data of the slave ADC. Refer to Section 27.4.31: Dual ADC modes.
The data alignment is applied as described in Section: Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.
In dual mode, these bits contain the regular data of the master ADC. Refer to Section 27.4.31: Dual ADC modes.
The data alignment is applied as described in Section: Data register, data alignment and offset (ADC_DR, OFFSET, OFFSET_CH, ALIGN)

In MDMA = 0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

27.8.4 ADC hardware configuration register (ADC_HWCFGR0)

Address offset: 0x3F0
Reset value: 0x0000 1112

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 15 | 14 | 13 | 12 | 11 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IDLEVALUE[3:0]**: Idle value for non-selected channels
0000: Dummy channel selection is 0x13
0001: Dummy channel selection is 0x1F
27.8.5 **ADC version register (ADC_VERR)**

Address offset: 0x3F4

Reset value: 0x0000 0013

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td>MAJREV[3:0]: Major revision</td>
</tr>
<tr>
<td></td>
<td>These bits returns the ADC IP major revision</td>
</tr>
<tr>
<td></td>
<td>0001: Major revision = 1.X</td>
</tr>
<tr>
<td>Bits 3:0</td>
<td>MINREV[3:0]: Minor revision</td>
</tr>
<tr>
<td></td>
<td>These bits returns the ADC IP minor revision</td>
</tr>
<tr>
<td></td>
<td>0001: Minor revision = X.1</td>
</tr>
<tr>
<td></td>
<td>0002: Minor revision = X.2</td>
</tr>
<tr>
<td></td>
<td>0003: Minor revision = X.3</td>
</tr>
</tbody>
</table>

27.8.6 **ADC identification register (ADC_IPDR)**

Address offset: 0x3F8

Reset value: 0x0011 0006

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>ID[31:16]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td><strong>ID[15:0]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
</tbody>
</table>
Bits 31:0 **ID[31:0]**: Peripheral identifier
   These bits returns the ADC identifier.
   \[\text{ID}[31:0] = 0x0011 \ 0006: \text{c7amba}_\text{aditf5}_90_v1\]

**27.8.7 ADC size identification register (ADC_SiDR)**

Address offset: 0x3FC

Reset value: 0xA3C5 DD01

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>SID[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>SID[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **SID[31:0]**: Size Identification
   SID[31:8]: fixed code that characterizes the ADC_SiDR register. This field is always read at 0xA3C5DD.
   SID[7:0]: read-only numeric field that returns the address offset (in Kbytes) of the identification registers from the IP base address:
   - 0x01: 1 Kbytes address offset
   - 0x02: 2 Kbytes address offset
   - 0x04: 4 Kbytes address offset
   - 0x08: 8 Kbytes address offset
27.9 ADC register map

The following table summarizes the ADC registers.

### Table 242. ADC global register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000 - 0x0B4</td>
<td>Master ADC1</td>
</tr>
<tr>
<td>0x0B8 - 0x0FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x100 - 0x1B4</td>
<td>Slave ADC2</td>
</tr>
<tr>
<td>0x1B8 - 0x2FC</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x300 - 0x30C</td>
<td>Master and slave ADCs common registers</td>
</tr>
</tbody>
</table>

### Table 243. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ADC_ISR</td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>ADC_IER</td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>ADC_CR</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFRG</td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_CFGR2</td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>ADC_SMPR1</td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>ADC_SMPR2</td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>ADC_TR1</td>
<td></td>
</tr>
</tbody>
</table>

Reset value columns contain hexadecimal values, where each column represents a bit in the register. For example, `0x00` indicates a reset value of 0, and `0x04` indicates a reset value of 0x04.
### Table 243. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register address</th>
<th>Offset CH[4:0]</th>
<th>SATEN</th>
<th>OFFSETPOS</th>
<th>OFFSET[11:0]</th>
<th>JEXTSEL</th>
<th>JL[1:0]</th>
<th>Offset Register name</th>
<th>Reset value</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x24</td>
<td>ADC_TR2</td>
<td></td>
<td>1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HT2[7:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x28</td>
<td>ADC_TR3</td>
<td></td>
<td>1 1 1 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>LT3[7:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x30</td>
<td>ADC_SQR1</td>
<td></td>
<td>SQ4[4:0]</td>
<td>0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>SQ1[4:0]</td>
<td>L[3:0]</td>
<td>SQ2[4:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x34</td>
<td>ADC_SQR2</td>
<td></td>
<td>SQ8[4:0]</td>
<td>0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>SQ7[4:0]</td>
<td>SQ6[4:0]</td>
<td>SQ5[4:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x38</td>
<td>ADC_SQR3</td>
<td></td>
<td>SQ14[4:0]</td>
<td>0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>SQ12[4:0]</td>
<td>SQ11[4:0]</td>
<td>SQ10[4:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x3C</td>
<td>ADC_SQR4</td>
<td></td>
<td>SQ16[4:0]</td>
<td>0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>SQ15[4:0]</td>
<td></td>
<td></td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x40</td>
<td>ADC_DR</td>
<td></td>
<td>JSQ4[4:0]</td>
<td>0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>JSQ3[4:0]</td>
<td>JSQ2[4:0]</td>
<td>JSQ1[4:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x50</td>
<td>ADC_OFR1</td>
<td></td>
<td>OFFSET_EN</td>
<td></td>
<td>OFFSETPOS</td>
<td>OFFSET[11:0]</td>
<td></td>
<td></td>
<td>OFFSET[11:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x54</td>
<td>ADC_OFR2</td>
<td></td>
<td>OFFSET_EN</td>
<td></td>
<td>OFFSETPOS</td>
<td>OFFSET[11:0]</td>
<td></td>
<td></td>
<td>OFFSET[11:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x58</td>
<td>ADC_OFR3</td>
<td></td>
<td>OFFSET_EN</td>
<td></td>
<td>OFFSETPOS</td>
<td>OFFSET[11:0]</td>
<td></td>
<td></td>
<td>OFFSET[11:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x5C</td>
<td>ADC_OFR4</td>
<td></td>
<td>OFFSET_EN</td>
<td></td>
<td>OFFSETPOS</td>
<td>OFFSET[11:0]</td>
<td></td>
<td></td>
<td>OFFSET[11:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>

**Reset value for master ADC:**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>ADC_TR1</td>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x04</td>
<td>ADC_SQR1</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>ADC_SQR2</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C</td>
<td>ADC_SQR3</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>ADC_SQR4</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x14</td>
<td>ADC_DR</td>
<td>0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

**Reset value for slave ADC:**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18</td>
<td>ADC_TR2</td>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x1C</td>
<td>ADC_TR3</td>
<td>1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x20</td>
<td>ADC_SQR1</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x24</td>
<td>ADC_SQR2</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x28</td>
<td>ADC_SQR3</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x2C</td>
<td>ADC_SQR4</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x30</td>
<td>ADC_DR</td>
<td>0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

**Reset value for regular RDATA[15:0]:**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x34</td>
<td>ADC_OFR1</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x38</td>
<td>ADC_OFR2</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x3C</td>
<td>ADC_OFR3</td>
<td>0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x40</td>
<td>ADC_OFR4</td>
<td>0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

1200/3791 RM0477 Rev 6
### Table 243. ADC register map and reset values for each ADC (offset = 0x000 for master ADC, 0x100 for slave ADC) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>JDATA1[15:0]</th>
<th>JDATA2[15:0]</th>
<th>JDATA3[15:0]</th>
<th>JDATA4[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x80</td>
<td>ADC_JDR1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x84</td>
<td>ADC_JDR2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x88</td>
<td>ADC_JDR3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x8C</td>
<td>ADC_JDR4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x90-0x9C</td>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xA0</td>
<td>ADC_AWD2CR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xA4</td>
<td>ADC_AWD3CR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xA8-0xAC</td>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xB0</td>
<td>ADC_DIFSEL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xB4</td>
<td>ADC_CALFACT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xBB-0xBC</td>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xC0</td>
<td>ADC_OR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xCC-0xFD</td>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 244. ADC register map and reset values (master and slave ADC common registers)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>slave ADC2</th>
<th>master ADC1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x300</td>
<td>ADC_CSR</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x304</td>
<td>Reserved</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x308</td>
<td>ADC_CCR</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Reset value 0000000000000000
Table 244. ADC register map and reset values (master and slave ADC common registers) (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30C</td>
<td>ADC_CDR</td>
<td>RDATA_SLV[15:0] RDATA_MST[15:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x310-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3F0</td>
<td>ADC_HWCGR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3F4</td>
<td>ADC_VERR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3F8</td>
<td>ADC_IPDR</td>
<td>ID[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3FC</td>
<td>ADC_SIDR</td>
<td>SID[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
28 Digital temperature sensor (DTS)

28.1 Introduction

The device embeds a sensor that converts the temperature into a square wave which frequency is proportional to the temperature. The frequency is measured either with the PCLK or the LSE clock.

28.2 DTS main features

The temperature sensor block main features are the following:

- Start of measurement triggered by software or 4 hardware sources
- Programmable sampling time to increase temperature measurement precision
- Counter synchronized on LSE or PCLK clock
- Temperature watchdog on low and high threshold
- Interrupt generation when the temperature is lower or higher than predefined thresholds and at the end of measurement.
- Asynchronous wakeup signal generation when the temperature is higher/lower than a predefined threshold (LSE mode only)
- Quick measurement using LSE clock
28.3 DTS functional description

28.3.1 DTS block diagram

The temperature sensor block diagram is shown in Figure 296.

Figure 296. Temperature sensor functional block diagram

![Temperature sensor functional block diagram](image)

28.3.2 DTS internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dts_lse</td>
<td>Digital input</td>
<td>LSE clock</td>
</tr>
<tr>
<td>dts_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
<tr>
<td>dts_it</td>
<td>Digital output</td>
<td>Temperature sensor interrupt</td>
</tr>
<tr>
<td>dts_wkup</td>
<td>Digital output</td>
<td>Temperature sensor wakeup</td>
</tr>
</tbody>
</table>
28.3.3 DTS block operation

The analog part of the temperature sensor outputs a frequency that is proportional to the absolute temperature (CLK_PTAT). The frequency measurement is based on the PCLK or the LSE clock.

Before each measurement, the temperature sensor performs a calibration of the frequency generation blocks.

28.3.4 Operating modes

Several operating modes can be selected by setting the REFCLK_SEL bit in Temperature sensor configuration register 1 (DTS_CFGR1):

- PCLK only (REFCLK_SEL = 0)
  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using PCLK clock

- PCLK and LSE (REFCLK_SEL = 1)
  The temperature sensor registers can be accessed. The interface can consequently be reconfigured and the measurement sequence is performed using the LSE clock.

- LSE only (REFCLK_SEL = 1) and PCLK OFF
  The registers cannot be accessed. The measurement can be performed using the LSE clock. This mode is used to exit from Sleep mode by using hardware triggers and the asynchronous interrupt line.

28.3.5 Calibration

The temperature sensor must run the calibration prior to any frequency measurement. The calibration is performed automatically when the temperature measurement is triggered except for quick measurement mode (Q_MEAS_OPT set to 1 in DTS_CFGR1).

28.3.6 Prescaler

When a calibration is ongoing, the counter clock must be slower than 1 MHz. This is achieved by the PCLK clock prescaler embedded in the temperature sensor.

During the temperature measurement period, the prescaler is bypassed.

- When PCLK is used as reference clock (REFCLK_SEL set to 0 in DTS_CFGR1), a prescaler is used. Its division ratio must be configured up to 127 (refer to the HSREF_CLK_DIV[6:0] register definition for the divider setting).

- When LSE is used as reference clock (REFCLK_SEL set to 1 in DTS_CFGR1), the timebase is equal to 2 LSE periods. In this case, no prescaler is used.
28.3.7 Temperature measurement principles

The analog part of temperature sensor outputs a signal (CLK_PTAT) which FM(T) frequency is temperature-dependent.

Either PCLK or LSE can be selected as reference clock (REF_CLK) through the REFCLK_SEL bit in DTS_CFG1.

The counting method depends on the REF_CLK frequency. This is due to the fact that two counters are implemented in the temperature sensor block:

- For low REF_CLK frequencies, a counting of FM(T) cycles is performed during one or several REF_CLK cycles.
- For high REF_CLK frequencies, a counting of REF_CLK cycles is performed during one or several FM(T) cycles.

This counter behavior is shown in Figure 297 and Figure 298.

**Figure 297. Method for low REF_CLK frequencies**

1. To increase the precision, FM(T) measurement can be done on several LSE periods.

**Figure 298. Method for high REF_CLK frequencies**

1. To increase the precision, PCLK measurement can be done on several FM(T) periods.

The counting result is stored in the DTS_DR register (see Temperature sensor data register (DTS_DR)).

Once the FM(T) frequency has been obtained, the corresponding temperature can be calculated by software using the following formula:

- When PCLK is used:

\[ T = T_0 + \left( \frac{F_{PCLK}}{\text{TS1_MFREQ} \times \text{TS1_SMP_TIME} - 100 \times \text{TS1_FMT0}} \right) \times \text{TS1_RAMP_COEFF} \]

where

- \( T_0 \) (factory calibration temperature) is equal to 30 °C.
- TS1_FMT0 is measured and stored in the DTS_T0VALR1 register. It is expressed in hundreds of Hertz.
TS1_RAMP_COEFF is measured during tests in factory and stored in DTS_RAMPVALR register. This value is expressed in Hz/°C.

- When the LSE clock is used

\[ T = T_0 + (F_{\text{LSE}} \cdot TS1_{\text{MFREQ}} \cdot TS1_{\text{SMP\_TIME}} - (100 \cdot TS1_{\text{FMT0}})) / TS1_{\text{RAMP\_COEFF}} \]

### 28.3.8 Sampling time

The sampling period can be increased to improve measurement accuracy. This is useful when the reference frequency (REF_CLK) is close to the FM(T) frequency. The default value is one REF_CLK cycle in LSE mode, and one FM(T) cycle in PCLK mode.

The sampling time is configured through TS1_SMP_TIME bits in DTS_CFGR1 register (see Table 246).

<table>
<thead>
<tr>
<th>TS1_SMP_TIME[3:0]</th>
<th>LSE or FM(T) clock cycle(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>0010</td>
<td>2</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
</tr>
<tr>
<td>0100</td>
<td>4</td>
</tr>
<tr>
<td>0101</td>
<td>5</td>
</tr>
<tr>
<td>0110</td>
<td>6</td>
</tr>
<tr>
<td>0111</td>
<td>7</td>
</tr>
<tr>
<td>1000</td>
<td>8</td>
</tr>
<tr>
<td>1001</td>
<td>9</td>
</tr>
<tr>
<td>1010</td>
<td>10</td>
</tr>
<tr>
<td>1011</td>
<td>11</td>
</tr>
<tr>
<td>1100</td>
<td>12</td>
</tr>
<tr>
<td>1101</td>
<td>13</td>
</tr>
<tr>
<td>1110</td>
<td>14</td>
</tr>
<tr>
<td>1111</td>
<td>15</td>
</tr>
</tbody>
</table>

### 28.3.9 Quick measurement mode

If a high precision is not required, the calibration step included in each measurement sequence can be skipped by setting Q_MEAS_OPT to 1 in the DTS_CFGR1 register. This method must be used only when the LSE clock is selected as reference clock (LSREF_CLK set to 1). This mode can reduce the measurement time.
28.3.10 Trigger input

The temperature measurement can be triggered either by software or by an external event. The trigger source can be selected through TS1_INTRIG[3:0] bits in DTS_CFRG1.

Table 247. Trigger configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>TS1_INTRIG[3:0]</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.A</td>
<td>0 0 0 0</td>
<td>No hardware trigger</td>
</tr>
<tr>
<td>ts1_trg0</td>
<td>0 0 0 1</td>
<td>lptim4_out</td>
</tr>
<tr>
<td>ts1_trg1</td>
<td>0 0 1 0</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>ts1_trg2</td>
<td>0 0 1 1</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>ts1_trg3</td>
<td>0 1 0 0</td>
<td>extl13</td>
</tr>
<tr>
<td>ts1_trg4</td>
<td>0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg5</td>
<td>0 1 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg6</td>
<td>0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg7</td>
<td>1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg8</td>
<td>1 0 0 1</td>
<td>Reserved</td>
</tr>
<tr>
<td>ts1_trg9</td>
<td>1 0 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg10</td>
<td>1 0 1 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg11</td>
<td>1 1 0 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg12</td>
<td>1 1 0 1</td>
<td></td>
</tr>
<tr>
<td>ts1_trg13</td>
<td>1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>ts1_trg14</td>
<td>1 1 1 1</td>
<td></td>
</tr>
</tbody>
</table>

Note: Hardware triggers are active only on the rising edge.

The temperature sensor can only capture a hardware trigger rising edge when TS1_RDY bit is set (see Section 28.3.11: On-off control and ready flag), otherwise the trigger is ignored. If a trigger source changes on-the-fly, the new trigger source signal should be low. If the new source signal is high, the temperature sensor detects a rising edge and starts the measurement sequence.

28.3.11 On-off control and ready flag

The DTS block can be enabled by setting TS1_EN bit in DTS_CFRG1 register. The TS1_RDY flag in the Temperature sensor status register (DTS_SR) indicate that the DTS block is ready for temperature measurement: when TS1_RDY bit is set to 1, the measurement can be started. Once a measurement has started, TS1_RDY bit is reset. New measurement requests are then ignored. Once the measurement is finished, TS1_RDY bit is set again to indicate the sensor is ready to start a new measurement.
28.3.12 Temperature measurement sequence

Start of measurement can be triggered by software or hardware.

Software trigger

The software trigger is selected when TS1_INTRIG_SEL[3:0] is set to '0000' in DTS_CFG1R.

If TS1_RDY is set to 1, writing TS1_START bit to 1 in DTS_CFG1R starts the measurement.
If TS1_RDY equals 0, the software trigger does not start until TS1_RDY is set.
If TS1_START bit is kept at 1 once the measurement is finished, then the TS1_RDY flag become 1 and the measurement restarts.

Hardware trigger

TS1_INTRIG_SEL[3:0] bits allow selecting one hardware trigger out of 4. If TS1_RDY is set to 1, a rising edge on the trigger signal starts the measurement. When TS1_RDY is 0, the rising edge is ignored.

Temperature measurement sequence

One measurement contains two steps: the calibration of the analog blocks and the measurement. The calibration automatically starts when the measurement is triggered (see Section 28.3.5: Calibration). The measurement period depends on the following DTS_CFG1R bits:

- the reference clock selected through REFCLK_SEL bit
- the divider ratio configured by HSREF_CLK_DIV bits
- the sampling time defined by TS1_SMP_TIME bits.

Figure 299. Temperature sensor sequence
28.4 DTS low-power modes

### Table 248. Temperature sensor behavior in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>Only works in LSE mode. DTS interrupt causes the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>Only works in LSE mode. DTS interrupt cause the device to exit from Stop mode.</td>
</tr>
</tbody>
</table>

28.5 DTS interrupts

There are two ways to use the DTS block as an interrupt source. The DTS interrupt line can be connected to the EXTI controller (see *Section 28.5.3: Asynchronous wakeup*) or to the CPU NVIC (see *Section 28.5.2: Synchronous interrupt*).

28.5.1 Temperature window comparator

The DTS_ITR1 register allows defining the high and low threshold that are used for temperature comparison. If the temperature data is equal or higher than TS1_HITTHD, or equal or lower than TS1_LITTHD bit, an interrupt is generated and the corresponding flag, TS1_ITHF, TS1_LTHF, TS1_AITLF and TS1_AITHF, is set in the DTS_SR register (see *Section 28.6.6*).

28.5.2 Synchronous interrupt

A global interrupt output line is available on the DTS block. The interrupt can be generated at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 28.5.1: Temperature window comparator*).

Three interrupt events can be select via 3 bits in DTS_ITENR register (see *Section 28.6.7*). All combinations of interrupts are allowed.

The TS1_ITEF, TS1_ITLF and TS1_ITHF flags in the DTS_SR register reflect the interrupt event. They can be reset with the correspond bits of the DTS_ICIFR register (see *Section 28.6.8*).

28.5.3 Asynchronous wakeup

The DTS block also provides an asynchronous interrupt line. It is used only when the LSE is selected as reference clock (REFCLK_SEL=1).

This line can generate a signal that wakes up the system from Sleep mode at the end of measurement and/or when the measurement result is equal/higher or equal/lower than a predefined threshold (see *Section 28.5.1: Temperature window comparator*).

Three asynchronous wakeup events can be selected via 3 bits in DTS_ITENR register. All combination of interrupts are allowed.
The TS1_AITEF, TS1_AITLF and TS1_AITHF flags in the DTS_SR register reflect the interrupt status. They can be reset with the correspond bits of the DTS_ICIFR register.

The following table shows the interrupt bits and their description.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Interrupt flag</th>
<th>Enable control bit</th>
<th>Interrupt clear bit</th>
<th>Exit from Sleep mode</th>
<th>Synchronous/Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the end of measurement</td>
<td>TS1_ITEF in DTS_SR</td>
<td>TS1_ITEEN in DTS_ITENR</td>
<td>TS1_CITEF in DTS_ICIFR</td>
<td>NO</td>
<td>Synchronous on PCLK</td>
</tr>
<tr>
<td>When the measure is equal or exceeds the low threshold</td>
<td>TS1_ITLF in DTS_SR</td>
<td>TS1_ITLEN in DTS_ITENR</td>
<td>TS1_CITLF in DTS_ICIFR</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>When the measure is equal or exceeds the high threshold</td>
<td>TS1_ITHF in DTS_SR</td>
<td>TS1_ITHEN in DTS_ITENR</td>
<td>TS1_CITHF in DTS_ICIFR</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>At the end of measurement</td>
<td>TS1_AITEF in DTS_SR</td>
<td>TS1_AITEEN in DTS_ITENR</td>
<td>TS1_CAITEF in DTS_ICIFR</td>
<td>YES</td>
<td></td>
</tr>
<tr>
<td>When the measure is equal or exceeds the low threshold</td>
<td>TS1_AITLF in DTS_SR</td>
<td>TS1_AITLEN in DTS_ITENR</td>
<td>TS1_CAITLF in DTS_ICIFR</td>
<td>YES</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>When the measure is equal or exceeds the high threshold</td>
<td>TS1_AITHF in DTS_SR</td>
<td>TS1_AITHEN in DTS_ITENR</td>
<td>TS1_CAITHF in DTS_ICIFR</td>
<td>YES</td>
<td></td>
</tr>
</tbody>
</table>

### 28.6 DTS registers

The registers of this peripheral can only be accessed by-word (32-bit).

#### 28.6.1 Temperature sensor configuration register 1 (DTS_CFGR1)

DTS_CFGR1 is the configuration register for temperature sensor 1.

Address offset: 0x00
Reset value: 0x0000 0000
Bit 31  Reserved, must be kept at reset value.

Bits 30:24  **HSREF_CLK_DIV[6:0]**: High speed clock division ratio
   These bits are set and cleared by software. They can be used to define the division ratio for
   the main clock in order to obtain the internal frequency lower than 1 MHz required for the
   calibration. They are applicable only for calibration when PCLK is selected as reference
   clock (REFCLK_SEL=0).
   0000000: No divider
   0000001: No divider
   0000010: 1/2 division ratio
   ...
   1111111: 1/127 division ratio

Bits 23:22  Reserved, must be kept at reset value.

Bit 21  **Q_MEAS_OPT**: Quick measurement option bit
   This bit is set and cleared by software. It is used to increase the measurement speed by
   suppressing the calibration step. It is effective only when the LSE clock is used as reference
   clock (REFCLK_SEL=1).
   0: Measurement with calibration
   1: Measurement without calibration

Bit 20  **REFCLK_SEL**: Reference clock selection bit
   This bit is set and cleared by software. It indicates whether the reference clock is the high
   speed clock (PCLK) or the low speed clock (LSE).
   0: High speed reference clock (PCLK)
   1: Low speed reference clock (LSE)

Bits 19:16  **TS1_SMP_TIME[3:0]**: Sampling time for temperature sensor 1
   These bits allow increasing the sampling time to improve measurement precision.
   When the PCLK clock is selected as reference clock (REFCLK_SEL = 0), the measurement
   is performed at TS1_SMP_TIME period of CLK_PTAT.
   When the LSE is selected as reference clock (REFCLK_SEL = 1), the measurement is
   performed at TS1_SMP_TIME period of LSE.

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

Bits 11:8  **TS1_INTRIG_SEL[3:0]**: Input trigger selection bit for temperature sensor 1
   These bits are set and cleared by software. They select which input triggers a temperature
   measurement. Refer to *Section 28.3.10: Trigger input*.

Bits 7:5  Reserved, must be kept at reset value.

Bit 4  **TS1_START**: Start frequency measurement on temperature sensor 1
   This bit is set and cleared by software.
   0: No software trigger.
   1: Software trigger for a frequency measurement. (only if TS1 is ready).

Bits 3:1  Reserved, must be kept at reset value.

Bit 0  **TS1_EN**: Temperature sensor 1 enable bit
   This bit is set and cleared by software.
   0: Temperature sensor 1 disabled
   1: Temperature sensor 1 enabled

*Note:* Once enabled, the temperature sensor is active after a specific delay time. The
**TS1_RDY** flag is set when the sensor is ready.
28.6.2 Temperature sensor T0 value register 1 (DTS_T0VALR1)

DTS_T0VALR1 contains the value of the factory calibration temperature (T0) for temperature sensor 1. The reset value is factory trimmed.

Address offset: 0x08
Reset value: 0x000X XXXX

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TS1_T0[1:0]

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

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

Bits 17:16 **TS1_T0[1:0]**: Engineering value of the T0 temperature for temperature sensor 1.

00: 30 °C
01: 130 °C
Others: Reserved, must not be used.

Bits 15:0 **TS1_FMT0[15:0]**: Engineering value of the frequency measured at T0 for temperature sensor 1. This value is expressed in 0.1 kHz.

28.6.3 Temperature sensor ramp value register (DTS_RAMPVALR)

The DTS_RAMPVALR is the ramp coefficient for the temperature sensor. The reset value is factory trimmed.

Address offset: 0x10
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TS1_RAMP_COEFF[15:0]

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bits 31:16: Reserved, must be kept at reset value.

Bits 15:0 **TS1_RAMP_COEFF[15:0]**: Engineering value of the ramp coefficient for the temperature sensor 1. This value is expressed in Hz/°C.
28.6.4 Temperature sensor interrupt threshold register 1 (DTS_ITR1)

DTS_ITR1 contains the threshold values for sensor 1.
Address offset: 0x14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

TS1_HITTHD[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

TS1_LITTHD[15:0]

Bits 31:16 TS1_HITTHD[15:0]: High interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the highest value than can be reached before raising an interrupt signal.

Bits 15:0 TS1_LITTHD[15:0]: Low interrupt threshold for temperature sensor 1
These bits are set and cleared by software. They indicate the lowest value than can be reached before raising an interrupt signal.

28.6.5 Temperature sensor data register (DTS_DR)

The DTS_DR contains the number of REF_CLK cycles used to compute the FM(T) frequency.
Address offset: 0x1C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

TS1_MFREQ[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 TS1_MFREQ[15:0]: Value of the counter output value for temperature sensor 1
28.6.6 Temperature sensor status register (DTS_SR)

Address offset: 0x20
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>


r r r r r r r r r r r r r r r r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TS1_RDY**: Temperature sensor 1 ready flag
- This bit is set and reset by hardware.
- It indicates that a measurement is ongoing.
  0: Temperature sensor 1 busy
  1: Temperature sensor 1 ready

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **TS1_AITHF**: Asynchronous interrupt flag for high threshold on temperature sensor 1
- This bit is set by hardware when the high threshold is reached.
- It is cleared by software by writing 1 to the TS1_CAITHF bit in the DTS_ICIFR register.
  0: High threshold not reached on temperature sensor 1
  1: High threshold reached on temperature sensor 1

*Note: This bit is active only when the TS1_AITHFEN bit is set*

Bit 5 **TS1_AITLF**: Asynchronous interrupt flag for low threshold on temperature sensor 1
- This bit is set by hardware when the low threshold is reached.
- It is cleared by software by writing 1 to the TS1_CAITLF bit in the DTS_ICIFR register.
  0: Low threshold not reached on temperature sensor 1
  1: Low threshold reached on temperature sensor 1

*Note: This bit is active only when the TS1_AITLFEN bit is set*

Bit 4 **TS1_AITEF**: Asynchronous interrupt flag for end of measure on temperature sensor 1
- This bit is set by hardware when a temperature measure is done.
- It is cleared by software by writing 1 to the TS1_CAITEF bit in the DTS_ICIFR register.
  0: End of measure not detected on temperature sensor 1
  1: End of measure detected on temperature sensor 1

*Note: This bit is active only when the TS1_AITEFEN bit is set*

Bit 3 Reserved, must be kept at reset value.
Bit 2 **TS1ITHF**: Interrupt flag for high threshold on temperature sensor 1, synchronized on PCLK.
   This bit is set by hardware when the high threshold is set and reached.
   It is cleared by software by writing 1 to the TS1_CITHF bit in the DTS_ICIFR register.
   0: High threshold not reached on temperature sensor 1
   1: High threshold reached on temperature sensor 1
   *Note: This bit is active only when the TS1ITHFEN bit is set*

Bit 1 **TS1ITLF**: Interrupt flag for low threshold on temperature sensor 1, synchronized on PCLK.
   This bit is set by hardware when the low threshold is set and reached.
   It is cleared by software by writing 1 to the TS1_CILTF bit in the DTS_ICIFR register.
   0: Low threshold not reached on temperature sensor 1
   1: Low threshold reached on temperature sensor 1
   *Note: This bit is active only when the TS1ITLFEN bit is set*

Bit 0 **TS1ITEF**: Interrupt flag for end of measurement on temperature sensor 1, synchronized on PCLK.
   This bit is set by hardware when a temperature measure is done.
   It is cleared by software by writing 1 to the TS2_CITEF bit in the DTS_ICIFR register.
   0: No end of measurement detected on temperature sensor 1
   1: End of measure detected on temperature sensor 1
   *Note: This bit is active only when the TS1ITEFEN bit is set*

### 28.6.7 Temperature sensor interrupt enable register (DTS_ITENR)

Address offset: 0x24

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1AITHEN**: Asynchronous interrupt enable flag on high threshold for temperature sensor 1.
   This bit is set and cleared by software.
   It enables the asynchronous interrupt when the temperature is above the high threshold
   (only when REFCLK_SEL = 1’b1)
   0: Asynchronous interrupt on high threshold disabled for temperature sensor 1
   1: Asynchronous interrupt on high threshold enabled for temperature sensor 1

Bit 5 **TS1AITLEN**: Asynchronous interrupt enable flag for low threshold on temperature sensor 1.
   This bit is set and cleared by software.
   It enables the asynchronous interrupt when the temperature is below the low threshold (only
   when REFCLK_SEL = 1)
   0: Asynchronous interrupt on low threshold disabled for temperature sensor 1
   1: Asynchronous interrupt on low threshold enabled for temperature sensor 1
Bit 4 **TS1_AITEEN**: Asynchronous interrupt enable flag for end of measurement on temperature sensor 1

This bit are set and cleared by software.
It enables the asynchronous interrupt for end of measurement (only when REFCLK_SEL = 1).
0: Asynchronous interrupt for end of measurement disabled on temperature sensor 1
1: Asynchronous interrupt for end of measurement enabled on temperature sensor 1

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TS1_ITHEN**: Interrupt enable flag for high threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the interrupt when the measure reaches or is above the high threshold.
0: Synchronous interrupt for high threshold disabled on temperature sensor 1
1: Synchronous interrupt for high threshold enabled on temperature sensor 1

Bit 1 **TS1_ITLEN**: Interrupt enable flag for low threshold on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the synchronous interrupt when the measure reaches or is below the low threshold.
0: Synchronous interrupt for low threshold disabled on temperature sensor 1
1: Synchronous interrupt for low threshold enabled on temperature sensor 1

Bit 0 **TS1_ITEEN**: Interrupt enable flag for end of measurement on temperature sensor 1, synchronized on PCLK.

This bit are set and cleared by software.
It enables the synchronous interrupt for end of measurement.
0: Synchronous interrupt for end of measurement disabled on temperature sensor 1
1: Synchronous interrupt for end of measurement enabled on temperature sensor 1

### 28.6.8 Temperature sensor clear interrupt flag register (DTS_ICIFR)

DTS_ICIFR is the control register for the interrupt flags.

Address offset: 0x28
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **TS1_CAITHF**: Asynchronous interrupt clear flag for high threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITHF flag in the DTS_SR register.

Bit 5 **TS1_CAITLF**: Asynchronous interrupt clear flag for low threshold on temperature sensor 1
Writing 1 to this bit clears the TS1_AITLF flag in the DTS_SR register.
Bit 4 **TS1_CAITEF**: Write once bit. Clear the asynchronous IT flag for End Of Measure for thermal sensor 1.

Writing 1 clears the TS1_AITEF flag of the DTS_SR register.

Bit 3 **Reserved**, must be kept at reset value.

Bit 2 **TS1_CITHF**: Interrupt clear flag for high threshold on temperature sensor 1

Writing this bit to 1 clears the TS1_IITHF flag in the DTS_SR register.

Bit 1 **TS1_CITLF**: Interrupt clear flag for low threshold on temperature sensor 1

Writing 1 to this bit clears the TS1_ITLF flag in the DTS_SR register.

Bit 0 **TS1_CITEF**: Interrupt clear flag for end of measurement on temperature sensor 1

Writing 1 to this bit clears the TS1_ITEF flag in the DTS_SR register.

### 28.6.9 Temperature sensor option register (DTS_OR)

The DTS_OR contains general-purpose option bits.

Address offset: 0x2C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TS_OP</strong></td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>---------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| **TS_OP** | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bits 31:0 **TS_OP[31:0]**: general purpose option bits
### 28.6.10 DTS register map

The following table summarizes the temperature sensor registers.

Table 250. DTS register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register name</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>DTS_CFGR1</td>
<td>HSREF_CLK_DIV</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x04</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>DTS_T0VALR1</td>
<td>TS1_T0[15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10</td>
<td>DTS_RAMPVALR</td>
<td>TS1_RAMP_COEFF[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x14</td>
<td>DTS_ITR1</td>
<td>TS1_HITTHD[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x18</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x1C</td>
<td>DTS_DR</td>
<td>TS1_MFREQ[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x20</td>
<td>DTS_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>DTS_ITEMR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x28</td>
<td>DTS_ICIFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>DTS_OR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
29 Voltage reference buffer (VREFBUF)

29.1 Introduction

The devices embed a voltage reference buffer which can be used as voltage reference for ADCs and also as voltage reference for external components through the VREF+ pin.

29.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFBUF0</td>
<td>2.5 V</td>
</tr>
<tr>
<td>VREFBUF1</td>
<td>2.048 V</td>
</tr>
<tr>
<td>VREFBUF2</td>
<td>1.8 V</td>
</tr>
</tbody>
</table>

Note: Refer to the product datasheet for more details.

29.3 VREFBUF functional description

The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference VREFINT. The VREFBUF supports three voltages\(^1\), which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected.
- VRS = 001: VREFBUF1 voltage selected.
- VRS = 010: VREFBUF2 voltage selected.

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:
After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

### 29.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

### 29.5 VREFBUF registers

#### 29.5.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The minimum VDDA voltage depends on VRS setting, refer to the product datasheet.
Bits 31:7  Reserved, must be kept at reset value.

Bits 6:4  **VRS[2:0]**: Voltage reference scale
These bits select the value generated by the voltage reference buffer.
- VRS = 000: VREFBUF0 voltage selected.
- VRS = 001: VREFBUF1 voltage selected.
- VRS = 010: VREFBUF2 voltage selected.
- Others: Reserved

*Note: Refer to the product datasheet for each VREFBUFx voltage setting value.*

The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3  **VRR**: Voltage reference buffer ready
- 0: the voltage reference buffer output is not ready.
- 1: the voltage reference buffer output reached the requested level.

Bit 2  Reserved, must be kept at reset value.

Bit 1  **HIZ**: High impedance mode
This bit controls the analog switch to connect or not the VREF+ pin.
- 0: VREF+ pin is internally connected to the voltage reference buffer output.
- 1: VREF+ pin is high impedance.

Refer to [Table 252: VREF buffer modes](#) for the mode descriptions depending on ENVR bit configuration.

Bit 0  **ENVR**: Voltage reference buffer mode enable
This bit is used to enable the voltage reference buffer mode.
- 0: Internal voltage reference mode disable (external voltage reference mode).
- 1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

### 29.5.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TRIM[5:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
29.5.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

| Offset | Register name   | VREFBUF_CSR | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | VREFBUF_CSR   |             | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|       | Reset value   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | VREFBUF_CCR   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|       | Reset value   |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 for the register boundary addresses.
30 Audio digital filter (ADF)

30.1 Introduction

The audio digital filter (ADF) is a high-performance module dedicated to the connection of external sigma-delta (ΣΔ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- metering

The ADF features one digital serial interface (SITF0) and one digital filter (DFLT0) with flexible digital processing options in order to offer up to 24-bit final resolution.

The ADF serial interface supports several standards allowing the connection of various ΣΔ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

The ADF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition, it is possible to insert a high-pass filter.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter bypass, filter order, decimation ratio. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A sound activity detector (SAD) is available for the detection of sounds or voice signals. The SAD is connected at the output of the DFLT0. Several parameters can be programmed in order to adjust properly the SAD to the sound environment. The SAD strongly reduces the power consumption by preventing the storage of samples into the system memory, as long as the observed signal does not match the programmed criteria.

The digital processing is performed using only the kernel clock. The ADF requests the bus interface clock (AHB clock) only when data must be transferred or when a specific event requests the attention of the system processor.

30.2 ADF main features

- AHB Interface
- 1 serial digital input:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ΣΔ modulators
- 1 flexible digital filter path including:
  - A MCIC filter configurable in Sinc⁴ or Sinc⁵ filter with an adjustable decimation ratio
  - A reshape filter to improve the out-of-band rejection and in-band ripple
- A high-pass filter to cancel the DC offset
- Gain control
- Saturation blocks

- Clock absence detector
- Sound activity detector
- 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay the selected bitstream
- One trigger input
- Autonomous functionality in Stop modes
- DMA can be used to read the conversion data
- Interrupts services

### 30.3 ADF implementation

<table>
<thead>
<tr>
<th>Mode or feature</th>
<th>ADF1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of filters (DFL Tx) and serial interfaces (SITFx)</td>
<td>1</td>
</tr>
<tr>
<td>ADF_CK/I0 connected to pins</td>
<td>-</td>
</tr>
<tr>
<td>Sound activity detection (SAD)</td>
<td>X</td>
</tr>
<tr>
<td>RXFIFO depth (number of 24-bit words)</td>
<td>4</td>
</tr>
<tr>
<td>ADC connected to ADCITF1</td>
<td>ADC1</td>
</tr>
<tr>
<td>ADC connected to ADCITF2</td>
<td>ADC2</td>
</tr>
<tr>
<td>Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)</td>
<td>-</td>
</tr>
<tr>
<td>Main path with CIC4, CIC5</td>
<td>X</td>
</tr>
<tr>
<td>Main path with CIC1,2, 3 or FastSinc</td>
<td>-</td>
</tr>
<tr>
<td>RSFLT, HPF, SAT, SCALE, DLY, Discard functions</td>
<td>X</td>
</tr>
<tr>
<td>Autonomous in Stop modes</td>
<td>-</td>
</tr>
</tbody>
</table>

1. ‘X’ = supported, ‘-’ = not supported.
30.4  **ADF functional description**

30.4.1  **ADF block diagram**

![ADF block diagram](MSv63650V2)

30.4.2  **ADF pins and internal signals**

**Table 255. ADF external pins**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF_SDI0</td>
<td>Input</td>
<td>Data signal from external sensors.</td>
</tr>
<tr>
<td>ADF_CCKy (y = 0,1)</td>
<td>Input/output</td>
<td>Clock outputs for external sensor, or common clock input from external sensors</td>
</tr>
</tbody>
</table>

**Table 256. ADF internal signals**

<table>
<thead>
<tr>
<th>Name</th>
<th>Signal type</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>adf_trgli</td>
<td>Input</td>
<td>Trigger inputs to control the acquisition (see Table 257: ADF trigger connections for details)</td>
</tr>
<tr>
<td>adf_trgO</td>
<td>Output</td>
<td>Trigger output for synchronizing with other MDF instances</td>
</tr>
<tr>
<td>adf_flt0_dma</td>
<td>Input/output</td>
<td>DMA request/acknowledge signals for the ADF processing chain.</td>
</tr>
<tr>
<td>adf_flt0_it</td>
<td>Output</td>
<td>Global interrupt signals</td>
</tr>
</tbody>
</table>

1226/3791  RM0477 Rev 6
The SITF0 input interface allows the connection of the external sensor to the digital filter via the bitstream matrix (BSMX). The SITF0 can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The data from the serial interface is routed to the filter in order to perform the PDM to PCM conversion and the sound activity detection.

The serial interface is enabled by setting the SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external $\Sigma\Delta$ modulator.

**Note:** Before enabling the serial interface, the user must insure that the adf_proc_ck is already enabled (see Section 30.4.5: Clock generator (CKGEN) for details).

The SITF0 is controlled via the ADF serial interface control register 0 (ADF_SITF0CR).

As shown in the Figure 302, ADF_CCK0 or ADF_CCK1 can be selected as clock source, in order to sample the incoming bitstream:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on the ADF_CCK0 or ADF_CCK1 pin.
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source is the clock directly provided by the CCKDIV to the ADF_CCK0 or ADF_CCK1 pin.

See Table 258 for additional information.
LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the use of an adf_proc_ck clock frequency, only two times bigger than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the ADF must provide the bitstream clock to the external sensors via ADF_CCK0 and ADF_CCK1 pins. The ADF receives the bitstream data via the serial data input ADF_SDI0.

For the SITF0, the application must select the same clock than the one provided to the external sensor (ADF_CCK0 or ADF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the adf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select ADF_CCK0 or ADF_CCK1 clock for the capture of the data received via the ADF_SDI0 pin.

The ADF can generate a clock to the sensors via ADF_CCK0 or ADF_CCK1 if needed.

For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITF0 always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bs0_f)
- bitstream received using the bitstream clock rising edge (bs0_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.
To properly synchronize/receive the data stream, the adf_proc_ck frequency must be adjusted according to the constraints listed in Table 259.

**Clock absence detection**

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] bitfield of the SITF0. If the ADF does not detect clock transitions for a duration of STH[4:0] x T_{adf_proc_ck}, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] bitfield is in the ADF serial interface control register 0 (ADF_SITF0CR).

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.
   - If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.
3. Read the CKABF flag:
   - If CKABF = 1, go back to step 2.
   - If CKABF = 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

*Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.*

**Manchester mode**

In Manchester coded format, the ADF receives data stream from the external sensor via the ADF_SDI0 pin only.

The ADF_CCK0 and ADF_CCK1 pins are not needed in this mode.
Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bs0_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

To decode the incoming Manchester stream, the user must program STH[4:0] in the ADF serial interface control register 0 (ADF_SITF0CR). The STH[4:0] bitfield is used by the SITF0 to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the ADF_SDI0 input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates that the data value changed. Figure 304 shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate ($T_{SYM}$) must respect the following formula:

$$(STH + 1) \times T_{adf_proc_ck} < T_{SYM} < (2 \times STH \times T_{adf_proc_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = \text{round}\left(\frac{(2 \times OVR) - 1}{3}\right)$$
where OVR represents the ratio between the adf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the adf_proc_ck clock must be adjusted according to the constraints listed in Table 259.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x Tadf_proc_ck, or when the SITF0 is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the ADF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the Figure 304) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR) by writing CKABF bit to 1. If the serial interface is not yet synchronized the hardware immediately set the CKABF flag to 1.

2. Read the CKABF flag:
   - If CKABF= 1, go back to step 1.
   - If CKABF = 0, the Manchester interface is synchronized and provides valid data.

**Programming example**

In the following example, the ADF kernel clock frequency (Fadf_ker_ck) is 100 MHz and the received Manchester stream is at about 6 MHz (FSYMB):

1. Provide a valid adf_proc_ck to the SITF0.
   The adf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).
   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, Fadf_proc_ck = 50 MHz (8.33 times higher than the Manchester symbol frequency).

2. Compute STH.
   OVR is given by: OVR = Fadf_proc_ck / FSYMB = 50 MHz / 6 MHz = 8.33.

   Then STH[4:0] = \( \lfloor \frac{(2 \times \text{round}(\frac{8.33}{3}) - 1)}{5} \rfloor \)

   The minimum allowed frequency for the Manchester stream is then:
   \( 1 / (2 \times \text{STH} \times T_{\text{adf_proc_ck}}) = 1 / (10 \times 20 \text{ ns}) = 5 \text{ MHz} \)

   The maximum allowed frequency for the Manchester stream is then:
   \( 1 / ((\text{STH}+1) \times T_{\text{adf_proc_ck}}) = 1 / (6 \times 20 \text{ ns}) = 8.33 \text{ MHz} \)

**30.4.4 ADC slave interface (ADCITF)**

The ADCs are not always connected to the ADF. Refer to Section 30.3 to check the situation for this product.

The ADF allows the connection of up to two ADCs to the filter path. For the filter, the DATSRC[1:0] bitfield in the ADF digital filter configuration register 0 (ADF_DFLT0CICR) allows the application to select data from the ADCs.
Warning: The ADF does not support receiving interleaved data from one of the ADCITF input.

30.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the ADF:

- AHB clock (adf_hclk) used for the register interface
- kernel clock (adf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the ADF_CCK0 and ADF_CCK1 pins. All those clocks are generated from the adf_ker_ck.

The processing clock (adf_proc_ck) is used to run all the signal processing and to re-sample the incoming serial or parallel stream.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the ADF_CCK0 and ADF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

The adf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All the bits and fields controlling the CKGEN are in the Section 30.8.2: ADF clock generator control register (ADF_CKGCR).
The trigger logic for CKGEN is handled by the block TRG_CK. As shown in Figure 364, the CCKDIV divider can be triggered on the rising or falling edge of an external trigger source. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the ADF_CCKy (y = 0,1) clock of several ADF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the ADF_CCK0 or ADF_CCK1 pin thanks to CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

**Table 258. Control of the common clock generation**

<table>
<thead>
<tr>
<th>CCKyEN</th>
<th>CCKyDIR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>The ADF_CCKy pin is in input. An external clock can be connected to the ADF_CCKy pin and used by the SITF0 in order to decode the serial stream</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>The ADF_CCKy pin is in output. No clock is generated. The ADF_CCKy pin is driven low.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The ADF_CCKy pin is in output. A clock is generated on the ADF_CCKy pin. The SITF0 can use this pin as clock source in order to decode the serial stream</td>
</tr>
</tbody>
</table>

1. The configuration with CCKyEN = 1 and CCKyDIR = 0 is not shown must be avoided (no interest).

**Note:** The adf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITF0 or DFLT0).

**CKGEN activation sequence example**

- Set CKGDEN to 0.
- Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
• Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0], TRGSENS, CCK1EN and CCK0EN.
• Set CKGDEN to 1.

When needed, at any moment, CCK[1:0]EN bitfield value can be changed without disabling the clock generator.

Clock frequency constraints

The table below shows the frequency constraints to receive and process properly the samples.

Note: The reshape filter (RSFLT) needs up to 24 cycles of adf_proc_ck clock to process one sample.

Table 259. Clock constraints with respect to the incoming stream(1)

<table>
<thead>
<tr>
<th>SITF0 mode</th>
<th>ADF clock constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With RSFLT disabled</td>
</tr>
<tr>
<td></td>
<td>FADF_CCKy max frequency limited to 5 MHz</td>
</tr>
<tr>
<td>LF_MASTER</td>
<td>Fadf_proc_ck &gt; 2 x FADF_CCKy and</td>
</tr>
<tr>
<td></td>
<td>Fadf_hclk ≥ Fadf_proc_ck</td>
</tr>
<tr>
<td>SPI</td>
<td></td>
</tr>
<tr>
<td>MASTER SPI</td>
<td>FADF_CKx max frequency limited to 25 MHz</td>
</tr>
<tr>
<td>SLAVE SPI</td>
<td>Fadf_proc_ck &gt; 4 x FADF_CCKy and</td>
</tr>
<tr>
<td></td>
<td>Fadf_hclk ≥ Fadf_proc_ck</td>
</tr>
<tr>
<td>Manchester</td>
<td>FSYMB max frequency limited to 20 MHz</td>
</tr>
<tr>
<td></td>
<td>Fadf_proc_ck &gt; 6 x FSYMB and</td>
</tr>
<tr>
<td></td>
<td>Fadf_hclk ≥ Fadf_proc_ck</td>
</tr>
</tbody>
</table>

1. FADF_CCKy represents the frequency of clock received via ADF_CCKy, or generated via ADF_CCKy. FSYMB represents the frequency of the received symbol rate for Manchester mode.

30.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from the serial interface SITF0 and provides the selected stream to the digital filter DFLT0.

As shown in the Figure 355, the SITF0 provides two bitstreams (bs0_r and bs0_f) to the BSMX.

The application to select the wanted stream via the ADF bitstream matrix control register 0 (ADF_BSMX0CR). This selection is intended to be static.
BSMX programming sequence example

The BSSEL[4:0] bitfield cannot be changed if the DFLT0 is enabled. The following steps are needed to change the value of BSMX:

- Set DFLTEN of DFLT0 to 0.
- Wait for BSMXACTIVE = 0.
- Program BSSEL[4:0].
- Set DFLTEN of DFLT0 to 1.

30.4.7 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- clock skipper delay (DLY)
- MCIC decimation filter that can be configured in Sinc⁴ or Sinc⁵
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- receive RXFIFO
The figure below shows the filter path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

**Figure 307. DFLT overview**

**Symbol remap and source selection**

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

**Programmable micro-delay control (DLY)**

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate must be applied to the incoming stream. This feature can be used when the ADF is synchronized with another MDF instance (if present in the product) for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is done by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

The figure below shows an example on how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another MDF instance (if present in the product).
The CIC of the ADF (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is built from input samples a[23:16] while the sample N+1 of CIC1 is built from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

**Note:** When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the DFLT0 is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the DFLT0 is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

When the ADF skipped the amount of wanted data strobes, then SKPBF goes back to 0.

3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the ADF is enabled. If the application performs a D1 delay followed by a D2 delay, then all other active filters are delayed by D1 + D2.

**Note:** If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

### Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a Sinc\(^N\) function, this is why they are often called Sinc filters.
The Sinc\(^N\) digital filter embedded into the ADF can be configurable in Sinc\(^4\) or Sinc\(^5\), according to CICMOD:

- If CICMOD[2:0] = 4, Sinc\(^4\) is selected.
- If CICMOD[2:0] = 5, Sinc\(^5\) is selected.

The filters have the following transfer function:

\[
H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N
\]

where \(N\) can be 4 or 5, and \(D\) is the decimation ratio. \(D\) is equal to MCICD+1.

**Figure 309. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16**

**CIC output data size**

The size of samples delivered by the CIC (\(D_{\text{CIC}}\)), depends on the following parameters:

- CIC order (\(N\))
- CIC decimation ratio (\(D\))
- data size of the input stream (\(D_{\text{SIN}}\))

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (\(D_{\text{CIC}}\)) according to the parameters above.
The decimation ratio can be adjusted from 2 to 512 for the CIC filter.

The table below gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: $DS_{IN} = 1$ bit for a serial bitstream, but can be up to 16 bits when coming from the ADCITF.

**Table 260. Data size according to CIC order and CIC decimation values**

<table>
<thead>
<tr>
<th>Decimation</th>
<th>Data size (bits) when $DS_{IN} = 1$ bit (data from SITF)</th>
<th>Data size (bits) when $DS_{IN} = 12$ bits (data from ADCITF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sinc$^4$</td>
<td>Sinc$^5$</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>13</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>19</td>
</tr>
<tr>
<td>16</td>
<td>17</td>
<td>21</td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>23</td>
</tr>
<tr>
<td>24</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>32</td>
<td>21</td>
<td>26</td>
</tr>
<tr>
<td>48</td>
<td>24</td>
<td>-</td>
</tr>
<tr>
<td>64</td>
<td>25</td>
<td>-</td>
</tr>
<tr>
<td>76</td>
<td>26</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: For a full-scale input signal, the decimation ratio must not exceed 76 for a Sinc$^4$ and 32 for a Sinc$^5$.

The LSB parts of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-of-band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

**Scaling (SCALE) and saturation (SAT)**

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB), and can be increased by up to 12 bits (+ 72.2 dB).
The gain is adjusted by the SCALE[5:0] bitfield in the **ADF digital filter configuration register 0 (ADF_DFLT0CICR)**.

SCALE[5:0] can be changed even if the DFLT0 is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of adf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

The table below shows the possible gain values.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x20</td>
<td>-48.2</td>
<td>0x2B</td>
<td>-14.5</td>
<td>0x06</td>
<td>+18.1</td>
<td>0x11</td>
<td>+51.7</td>
</tr>
<tr>
<td>0x21</td>
<td>-44.6</td>
<td>0x2C</td>
<td>-12.0</td>
<td>0x07</td>
<td>+21.6</td>
<td>0x12</td>
<td>+54.2</td>
</tr>
<tr>
<td>0x22</td>
<td>-42.1</td>
<td>0x2D</td>
<td>-8.5</td>
<td>0x08</td>
<td>+24.1</td>
<td>0x13</td>
<td>+57.7</td>
</tr>
<tr>
<td>0x23</td>
<td>-38.6</td>
<td>0x2E</td>
<td>-6.0</td>
<td>0x09</td>
<td>+27.6</td>
<td>0x14</td>
<td>+60.2</td>
</tr>
<tr>
<td>0x24</td>
<td>-36.1</td>
<td>0x2F</td>
<td>-2.5</td>
<td>0x0A</td>
<td>+30.1</td>
<td>0x15</td>
<td>+63.7</td>
</tr>
<tr>
<td>0x25</td>
<td>-32.6</td>
<td>0x00</td>
<td>0.0</td>
<td>0x0B</td>
<td>+33.6</td>
<td>0x16</td>
<td>+66.2</td>
</tr>
<tr>
<td>0x26</td>
<td>-30.1</td>
<td>0x01</td>
<td>+3.5</td>
<td>0x0C</td>
<td>+36.1</td>
<td>0x17</td>
<td>+69.7</td>
</tr>
<tr>
<td>0x27</td>
<td>-26.6</td>
<td>0x02</td>
<td>+6.0</td>
<td>0x0D</td>
<td>+39.6</td>
<td>0x18</td>
<td>+72.2</td>
</tr>
<tr>
<td>0x28</td>
<td>-24.1</td>
<td>0x03</td>
<td>+9.5</td>
<td>0x0E</td>
<td>+42.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0x29</td>
<td>-20.6</td>
<td>0x04</td>
<td>+12.0</td>
<td>0x0F</td>
<td>+45.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0x2A</td>
<td>-18.1</td>
<td>0x05</td>
<td>+15.6</td>
<td>0x10</td>
<td>+48.2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The ADF performs saturation operations at the following levels:
- after the SCALE block (performed by the SAT block): The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the **ADF DFLT0 interrupt enable register (ADF_DFLT0IER)** and **ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)**.

**Gain adjustment policy**

To get the best ADF performances, it is important to properly adjust the gain value via SCALE[5:0].
A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full-scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

\[
\frac{\ln(\sin_{pp} \cdot D^N)}{\ln(2)} < 26
\]

where \( N \) represents the CIC order, \( D \) the decimation ratio and \( \sin_{pp} \) the maximum peak-to-peak amplitude of the input signal.

\( \sin_{pp} \) can take:
- a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF0
- A maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048), for samples coming from a 12-bit ADC

**Example:** a Sinc^4 can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

\[
\frac{\ln(0.7 \cdot 96^4)}{\ln(2)} = -25.82 \text{ bits} < 26 \text{ bits}
\]
2. Adjust the SCALE value.

To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

The SCALE value in dB must be selected using this formula:

\[
\text{SCALE}_{\text{dB}} < 20 \cdot \log_{10}\left( \frac{2^{\text{NB}}}{\text{SIN}_{\text{pp}} \cdot \text{D}^{N}} \right)
\]

where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. SCALE\text{dB} represents the gain value selected by SCALE[5:0].

Example: For a Sinc\(^4\) with a decimation ratio of 96 and a SIN\(_{pp}\) of 0.7.

- If the RSFLT is bypassed:

\[
20 \cdot \log_{10}\left( \frac{2^{24}}{0.7 \cdot 96^4} \right) = -11 \text{ dB}
\]

SCALE\text{dB} value must be lower than -11 dB, the closest lower value is -12 dB (SCALE[5:0] = 0x2C).

- If the RSFLT is enabled:

\[
20 \cdot \log_{10}\left( \frac{2^{22}}{0.7 \cdot 96^4} \right) = -23 \text{ dB}
\]

SCALE\text{dB} value must be lower than -23 dB, the closest lower value is -24.1 dB (SCALE[5:0] = 0x28).

If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

The table below proposes gain values for different filter configurations, when the data comes from the SITF0, according to the MCIC order, and the MCIC decimation ratio. This table is not exhaustive and considers a full-scale input signal (see Section 40.7.5: Total ADF gain for details).

<table>
<thead>
<tr>
<th>CIC decimation ratio</th>
<th>Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)</th>
<th>Gain settings (dB) for configuration SITF + CICx (+ HPF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIC5</td>
<td>CIC4</td>
</tr>
<tr>
<td>8</td>
<td>33.6</td>
<td>51.7</td>
</tr>
<tr>
<td>12</td>
<td>18.1</td>
<td>39.6</td>
</tr>
<tr>
<td>16</td>
<td>3.5</td>
<td>27.6</td>
</tr>
<tr>
<td>20</td>
<td>-6.0</td>
<td>21.6</td>
</tr>
<tr>
<td>24</td>
<td>-12.0</td>
<td>15.6</td>
</tr>
<tr>
<td>28</td>
<td>-20.6</td>
<td>9.5</td>
</tr>
</tbody>
</table>
Reshaping filter (RSFLT)

In addition to the CIC, the ADF offers a reshaping IIR filter mainly dedicated to the audio application, but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTDto bit in the ADF reshape filter configuration register 0 (ADF_DFLT0RSFR).

The RSFLT can be bypassed by setting RSFBYP to 1 in the ADF reshape filter configuration register 0 (ADF_DFLT0RSFR).

The table below shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency ($F_C$) depends on the sample rates at its input ($F_{RS}$), and is given by the following formula:

$$F_C = 0.111 \times F_{RS}$$

### Table 262. Recommended maximum gain values versus CIC decimation ratios

<table>
<thead>
<tr>
<th>CIC decimation ratio</th>
<th>Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)</th>
<th>Gain settings (dB) for configuration SITF + CICx (+ HPF)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CIC5</td>
<td>CIC4</td>
</tr>
<tr>
<td>32</td>
<td>- 26.6</td>
<td>3.5</td>
</tr>
<tr>
<td>48</td>
<td>-</td>
<td>- 8.5</td>
</tr>
<tr>
<td>64</td>
<td>-</td>
<td>- 20.6</td>
</tr>
</tbody>
</table>

### Table 263. Most common microphone settings

<table>
<thead>
<tr>
<th>Sample rate (kHz) at RSFLT ($F_{RS}$)</th>
<th>Pass band (kHz)</th>
<th>D2</th>
<th>PCM sampling rate (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>3.55</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>64</td>
<td>7.1</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>128</td>
<td>14.2</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>192</td>
<td>21.3</td>
<td>4</td>
<td>48</td>
</tr>
</tbody>
</table>
The figure below shows the frequency response of the reshape filter.

Figure 310. Reshape filter frequency response normalized (FRS / 2 = 1)

The RSFLT gain is close to 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of adf_proc_ck clock to process one sample at FRS. When the RSFLT is enabled, the application must insure that the adf_proc_ck is at least 24 times faster FRS.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new samples while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provides a faster adf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the ADF reshape filter configuration register 0 (ADF_DFLT0RSFR).

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source that must be removed from the final data.
The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the ADF reshape filter configuration register 0 (ADF_DFLT0RSFR), among the following values:

- 0.000625 x FPCM
- 0.00125 x FPCM
- 0.00250 x FPCM
- 0.00950 x FPCM

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

### 30.4.8 Digital filter acquisition modes

The ADF offers the following modes to perform a data capture:

- asynchronous continuous acquisition mode
- asynchronous single-shot acquisition mode
- synchronous continuous acquisition mode
- synchronous single-shot acquisition mode
- window continuous acquisition mode

**Note:** To perform a data capture, the filter, the interface providing the data (SITF0 or ADCITF) and the CKGEN must be enabled. If needed, the ADF_CCK0 or ADF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This action resets the filter and flushes the RXFIFO. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filter is reset.

<table>
<thead>
<tr>
<th>HPFC</th>
<th>3 dB cut-off frequency for common FPCM frequencies (Hz)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FPCM = 8 kHz</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>76</td>
</tr>
</tbody>
</table>
The figure below shows a simplified view of the trigger logic available for each filter and for the clock generator.

**Figure 311. Trigger logic for DFLT and CKGEN**

A block common to all TRIG blocks performs the rising and falling edges detection and the re-synchronization of the input trigger to the adf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the adf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select adf_trgi or TRGO bit as trigger sources. The TRGO bit is in the *ADF global control register (ADF_GCR)*.

The edge sensitivity can also be selected.

**Asynchronous continuous acquisition mode**

This mode allows the application to start a continuous acquisition by simply writing the DFLTEN bit to 1.

The asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN) so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 398*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the serial data interface.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the acquisition sequence starts immediately.
The figure below shows a simplified example of the samples generated by the DFLT0.

**Figure 312. Asynchronous continuous mode (ACQMOD[2:0] = 0)**

![Diagram](Image)

**Note:** The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

### Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample by simply writing the DFLTEN bit to 1.

The asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the adf_proc_ck frequency is compatible with the targeted application (see examples in Table 398).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

This sequence can be repeated every time a new data must be converted.

As shown in the Figure 366, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.
If DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

**Figure 313. Asynchronous single-shot mode (ACQMOD[2:0] = 001)**

Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

**Synchronous continuous acquisition mode**

This mode allows the application to start a continuous acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in Table 398).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 010.
4. Set to 1 the bit SITFEN.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition.
The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. Then the next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

**Figure 314. Synchronous continuous mode (ACQMOD[2:0] = 010)**

Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

**Synchronous single-shot acquisition mode**

This mode allows the application to start a single acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in Table 398).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition and provides one data to the RXFIFO, then the filter is ready to accept a new trigger.

TRGSSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing, or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSSENS = 0). Every-time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

**Figure 315. Synchronous single-shot mode (ACQMOD[2:0] = 011)**

Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

**Figure 368** shows a case where the DFLTEN is set to 0 while an acquisition is ongoing (the sample S2 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

```
<table>
<thead>
<tr>
<th>DFLTEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>adf_trgi</td>
</tr>
<tr>
<td>DFLT0 output</td>
</tr>
<tr>
<td>DFLTRUN</td>
</tr>
<tr>
<td>DFLTACTIVE</td>
</tr>
<tr>
<td>ADF_CCKx</td>
</tr>
</tbody>
</table>
```

Note: The discard phase is optional.
Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition controlled by consecutive edges of one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in Table 398).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 369 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the ADF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the ADF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the ADF waits again for a falling edge on the selected trigger input.

Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.
Starting several filters synchronously

If the ADF is used with MDF instances (if present in the product), it is possible to start simultaneously the acquisition of all the filters. This synchronization capability depends on the way the triggers are connected in the product. Generally, an ADF is able to trigger MDF instances, if its adf_trgo signal is connected as trigger input to those blocks (see Section 40.4.2: ADF pins and internal signals to check trigger capabilities).

In the following programing example, one ADF has its adf_trgo signal connected to some MDFs. To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN bits of the filters are set to 0):

On MDFs receiving the adf_trgo trigger:
1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] in order to select the adf_trgo trigger input.
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.

On the ADF generating the adf_trgo trigger:
1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK[1:0] clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. Set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. Set TRGSRC[3:0] to 0 (TRGO selected).
5. Set TRGSENS to 0 (rising edge).
6. Set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot) the application must do the following:
- Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
- Read TRGO until it is read to 0.
- Set again the bit TRGO to 1.

Discarded samples

The ADF offers the possibility to program the amount of samples to be discarded after each restart:
- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

The discard function is controlled via NBDIS[7:0] as follows:
- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to Figure 365 to Figure 369, and Figure 371.
In the example shown in the figure below, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transferred to the RXFIFO is S6.

**Figure 317. Discard function example**

### 30.4.9 Start-up sequence examples

The figure below details a start of acquisition sequence of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLT0 is configured for audio application: MCIC, RSFLT and HPF activated. The data interface (SITF0 or ADCITF) is assumed to be already activated.

**Note:** NBDIS[7:0] is set on purpose to a small value to simplify the drawing.

**Figure 318. Start sequence with DFLTEN, in continuous mode, audio configuration**
The DFLTEN bit is re-sampled into the ADF processing clock domain. When DFLTEN is detected high, the filter chain is enabled, and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reached its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC, and delivers a sample to the HPF every-time it processes four samples (decimation by 4). The RSFLT needs up to 24 cycles of adf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

### 30.4.10 Sound activity detection (SAD)

The SAD is based on the computation of the ambient noise level (ANLVL) and of the short-term sound level (SDLVL). The SAD offers the following ways to detect a sound:

- when the SDLVL reaches a threshold referenced to the ambient noise level
- when the SDLVL reaches a fixed threshold
- when the ANLVL reaches a fixed threshold

As shown in the figure below, the SAD takes the 16 MSB samples from the DFLT0 output.

**Figure 319. SAD block diagram**

The SAD is highly configurable, and the application can adjust several parameters:

- SAD detection behavior (SADMOD)
- number of samples used to compute the sound level (FRSIZE)
- number of frames used to compute the ambient noise level during the learning phase (LFRNB)
- slope of the ambient noise estimator (ANSLP)
- minimum expected ambient noise level (ANMIN)
- threshold level (SNTHR)
- threshold hysteresis (HYSTEN)
- hangover window in order to filter spurious transitions between DETECT and MONITOR states (HGOVR)
- data capture mode (DATCAP)

SAD detection behavior

The SAD can use the following ways to detect a sound, selected by SADMOD[1:0]:
- When SADMOD[1:0] = 0, the SAD works like a voice-activity detection. In this mode, the SAD estimates the ambient noise level according to the computed sound level values. The threshold of the trigger is elaborated from the estimated ambient noise. Finally the current sound level is compared to this threshold. In a first approximation, the SAD triggers if the peak-to-average value of the input signal reaches a level defined by SNTHR[3:0].
- When SADMOD[1:0] = 01, the SAD compares the current sound level (SDLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode allows a fast SAD reaction as the amount of samples used to compute the sound level can be configured via FRSIZE[2:0].
- When SADMOD[1:0] = 1x, the SAD compares the estimated ambient noise level (ANLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode avoids unwanted triggers, due to peak levels, but the SAD reacts more slowly to an input signal variation. It is nevertheless possible to adjust the reaction time via FRSIZE[2:0] and ANSLP[2:0].

SAD states

As shown in Figure 373, the SAD works as follows:

1. When enabled (SADEN = 1), the SAD is first in LEARN state to perform a first estimation of the ambient noise level.
2. The SAD continuously computes the short-term sound level (SDLVL) using the samples provided by the DFLT0. The amount of samples used to compute the sound level is given by FRSIZE[2:0]. The samples processed by the DFLT0 can be transferred into the memory or not depending on DATCAP[1:0] value.
3. The initial ambient noise level (ANLVL) is computed using the consecutive sound level values. The application can define how much sound level values are used to perform the computation of this initial ambient noise estimation (LFRNB).
4. When the initial ambient noise level (ANLVL) is computed, the SAD switches to the MONITOR state.
5. Every time a new short-term sound level value is available, the SAD updates the ambient noise level and the thresholds according to the selected detection mode.
6. If the SAD triggers, then the following happens:
   - The SAD switches to DETECT state.
   - The sddet_evt event is asserted.
   - The adf_sad_det signal is set to high.
7. The hangover function insures that the DETECT state is maintained even if the sound level goes below the threshold level for a time given by HGOVR.
Sound level computation (SDLVL)

Once enabled, the SAD computes continuously the sound level value. The sound level represents the average of the absolute value of an amount of PCM samples given by FRSIZE[2:0].

\[
SDLVL = \frac{1}{N_{FRSIZE}} \times \sum_{n=1}^{N_{FRSIZE}} |PCM(n)|
\]

where \(N_{FRSIZE}\) is the amount of PCM samples given by FRSIZE[2:0].
Ambient noise estimation (ANLVL)

The ambient noise level (ANLVL) is computed when SADMOD[1:0] is 00 or 10.

The ambient noise level is computed differently according to the state of the SAD as detailed below:

- **ANLVL computation during the LEARN state**
  
  Every time the SAD is enabled, a learning phase is initiated in order to estimate a first value of the ambient noise level. During this phase, the SAD cannot trigger.

  During the LEARN phase, the ambient noise level is computed as follows:

  \[
  \text{ANLVL} = \frac{1}{N_{\text{LFRNB}}} \sum_{n=1}^{N_{\text{LFRNB}}} |\text{SDLVL}(n)|
  \]

  where \( N_{\text{LFRNB}} \) is the amount of frames given by LFRNB[2:0] bitfield.

- **ANLVL computation during the MONITOR or DETECT state**

  When the learning phase is completed, the SAD updates the ambient noise level in the following way:

  a) The SAD computes the new possible values for the ambient noise level:

     \[
     \text{ANLVL\_UP} = \text{ANLVL} \times (1 + 2^{\text{ANSLP}-12})
     \]

     \[
     \text{ANLVL\_DN} = \text{ANLVL} \times (1 - 2^{\text{ANSLP}-10})
     \]

  b) The ANLVL takes the ANLVL\_DN value if the current sound level is lower than ANLVL\_DN, otherwise ANLVL takes the value of ANLVL\_UP.

     The ANLVL is not updated if the current sound level is higher than the threshold level, except if SADMOD[1:0] = 10.

  c) When SADMOD[1:0] = 0, if the new ANLVL value is lower than ANMIN[12:0], ANLVL is replaced by ANMIN.

The slope of the noise estimator can be adjusted to optimize the detection of the wanted signal. This slope is adjusted via ANSLP[2:0] in the ADF SAD configuration register (ADF_SADCFG).
The table below shows the allowed values according to the frame size and the sampling rate of the data observed by the SAD. The recommended values when the SADMOD[1:0] = 0 are the ones into the gray shaded cells.

**Table 265. ANSLP values versus FRSIZE and sampling rates**

<table>
<thead>
<tr>
<th>FRSIZE</th>
<th>ANSLP values for Fs = 8 kHz</th>
<th>ANSLP values for Fs = 16 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Slow(1) typical(2) Fast(3)</td>
<td>Slow(1) typical(2) Fast(3)</td>
</tr>
<tr>
<td>0 (8 samples)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 (16 samples)</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2 (32 samples)</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3 (64 samples)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4 (128 samples)</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5 (256 samples)</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>6 (512 samples)</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

1. The slow slope is equal to - 8.5 dB/s for the negative slope and + 2.1 dB/s for the positive slope.
2. The typical slope is equal to - 17.1 dB/s for the negative slope and + 4.2 dB/s for the positive slope.
3. The fast slope is equal to - 34.2 dB/s for the negative slope and + 8.5 dB/s for the positive slope.

The slopes can also be computed using the following formulas:

\[
SLC_{UP} = 20 \times \frac{F_s}{FSIZE} \times \log_{10}(1 + 2^{(ANSLP - 12)})
\]

\[
SLC_{DN} = 20 \times \frac{F_s}{FSIZE} \times \log_{10}(1 - 2^{(ANSLP - 10)})
\]

where \( F_s \) is the sampling rate of the stream observed by the SAD and \( F_{SIZE} \) is the frame size defined by \( FRSIZE[2:0] \).

**Threshold computation**

The way the threshold value is computed depends on SADMOD[1:0]:

- If SADMOD[1:0] = 0, THRH is obtained by multiplying the current ANLVL value with the gain defined in SNTHR[3:0].

\[
THR_H = ANLVL \times \frac{GdB_{SNTHR}}{20}
\]

This threshold value is then compared to the current sound level (SDLVL).

- If SADMOD[1:0] = 01, THRH is obtained by multiplying the current ANMIN[12:0] with the gain defined by SNTHR[3:0].

\[
THR_H = ANMIN \times \frac{GdB_{SNTHR}}{20}
\]

This threshold value is then compared to the current sound level (SDLVL).
If SADMOD[1:0] = 1x, THRH is obtained by multiplying the current ANMIN[12:0] by 4.

\[
\text{THR}_H = \text{ANMIN} \times 4
\]

This threshold value is then compared to:

\[
\frac{\text{ANLVL} \times 10^{\frac{\text{SNTHR}}{20}}}{G_{\text{dB}}}
\]

The hysteresis mode can be enabled to reduce the spurious transitions between MONITOR and DETECT states. In hysteresis mode (HYSTEN = 1), the following threshold values are used:

- THRH when the SAD is in MONITOR state.
- THRL when the SAD is in DETECT state.

The table below shows the thresholds values according to SNTHR.

<table>
<thead>
<tr>
<th>SNTHR[3:0]</th>
<th>THRH</th>
<th>THRL</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LVL + 3.5 dB</td>
<td>LVL x 1.5</td>
<td>LVL + 3.5 dB</td>
</tr>
<tr>
<td>1</td>
<td>LVL + 6.0 dB</td>
<td>LVL x 2</td>
<td>LVL + 3.5 dB</td>
</tr>
<tr>
<td>2</td>
<td>LVL + 9.5 dB</td>
<td>LVL x 3</td>
<td>LVL + 6.0 dB</td>
</tr>
<tr>
<td>3</td>
<td>LVL + 12.0 dB</td>
<td>LVL x 4</td>
<td>LVL + 9.5 dB</td>
</tr>
<tr>
<td>4</td>
<td>LVL + 15.6 dB</td>
<td>LVL x 6</td>
<td>LVL + 12.0 dB</td>
</tr>
<tr>
<td>5</td>
<td>LVL + 18.1 dB</td>
<td>LVL x 8</td>
<td>LVL + 15.6 dB</td>
</tr>
<tr>
<td>6</td>
<td>LVL + 21.6 dB</td>
<td>LVL x 12</td>
<td>LVL + 18.1 dB</td>
</tr>
<tr>
<td>7</td>
<td>LVL + 24.1 dB</td>
<td>LVL x 16</td>
<td>LVL + 21.6 dB</td>
</tr>
<tr>
<td>8</td>
<td>LVL + 27.6 dB</td>
<td>LVL x 24</td>
<td>LVL + 24.1 dB</td>
</tr>
<tr>
<td>9</td>
<td>LVL + 30.1 dB</td>
<td>LVL x 32</td>
<td>LVL + 27.6 dB</td>
</tr>
</tbody>
</table>

1. LVL must be replaced by ANLVL when SADMOD[1:0] = 0 and by ANMIN for other SADMOD[1:0] values.

When the hysteresis function is disabled, the SAD always use THRH.

**Note:**  
*The hysteresis mode must not be used when SADMOD[1:0] = 1x.*

**Trigger logic**

The signal compared to this threshold depends also on SADMOD[1:0].

The trigger condition is reached when the selected signal (SELSIG) is bigger than the threshold level.

If the trigger condition is met, the following happens:

- The SAD switches to DETECT state.
- The SAD refreshes the hangover counter with HGOVR.
- The sddet_evt event is asserted if the SAD transits from MONITOR to DETECT.
• The adf_sad_det signal is set to high.

The SAD remains in DETECT state as long as the trigger condition is met or the hangover
down-counter is different from 0.

The sddet_evt event indicates when the SAD enters and/or exits the DETECT state. This
event is used to generate an interrupt when a sound is detected or when a sound is no
longer detected:
• When DETCFG = 0, the application receives an event only when the SAD enters the
DETECT state.
• When DETCFG = 1, the application receives an event when the SAD enters or exits the
DETECT state.

The adf_sad_det signal remains high as long as the SAD is in DETECT state.

The SAD also provides a flag indicating that a new sound level value is available (SDLVLF).
The last computed sound level (SDLVL[14:0]) is available in the ADF SAD sound level
register (ADF_SADSDLVR), and the last computed ambient noise level (ANLVL[14:0]), in
the ADF SAD ambient noise level register (ADF_SADANLVR).

Note: The SAD can work even when the AHB clock is not present. In that case, the SAD does not
update SDLVL[14:0] and ANLVL[14:0].

To get the latest valid SDLVL[14:0] and ANLVL[14:0] values, the application must read the
ADF_SADSDLVR, and ADF_SADANLVR registers, when the SDLVLF flag goes high. This
can be done in the following ways:
• by polling the SDLVLF flag:
  a) Clear the SDLVLF flag by writing SDLVLF to 1.
  b) Wait for SDLVLF = 1, by reading ADF_DFLTxISR.
  c) Read ADF_SADSDLVR and ADF_SADANLVR.
  d) Clear SDLVLF by writing it to 1.
  e) Go to step 2 if other values must to be read.
• by generating an interrupt:
  a) Read ADF_DFLTxISR.
  b) If SDLVLF = 1, read ADF_SADSDLVR and ADF_SADANLVR, and clear SDLVLF
     by writing it to 1.
  c) Handle other status flags and exit from ISR.

Sample transfer to memory

The SAD offers the following options to control the samples transfer from DFLT0 to the
system memory:
• If DATCAP[1:0] = 1x, the samples are transferred into the system memory as soon as
  DFLT0 and SAD are enabled. The transfer does not depend on the SAD state.
• If DATCAP[1:0] = 01, the samples are transferred into the system memory when the
  SAD detects a sound (when the SAD is in DETECT state), assuming that DFLT0 and
  SAD are enabled.
• If DATCAP[1:0] = 0, the samples are not transferred into the memory. This mode can
  be used if the application only wants to observe but does not need samples for other
  processing.
Note: DATCAP[1:0] is taken into account only when the SADEN = 1. For example, if the SAD configuration is DATCAP[1:0] = 0, SADEN = DFLTEN = 1, and if the application sets now SADEN to 0, the samples provided by the DFLT0 are transferred to the RXFIFO.

Programming recommendations

To make the SAD function working properly, the ADF must be programmed as follows:

1. Provide the proper kernel clock (adf_ker_ck) to the ADF.
2. Configure the CKGEN and enable it.
3. Configure the SITF and enable it (note that microphones have a settling time of several milliseconds).
4. Configure the DFLT0. A typical setting is the following:
   - CIC5 with a decimation ratio of 12, 16 or 24
   - RSFLT with a decimation ratio of 4
   - HPF with HPFC = 2 or 3
   For a very-low power implementation, the RSFLT can be bypassed.
5. Set SADEN to 0.
6. Wait for SADACTIVE = 0. If SADEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
7. Configure the SAD as follows:
   - Set DATCAP[1:0] to 0, if the application does not want to store the samples into the system memory.
   - Set DATCAP[1:0] to 01, if the application wants to store the samples into the system memory only when the SAD detects a sound.
   - Set DATCAP[1:0] to 11, if the application wants to store the samples into the system memory continuously.
8. Configure the DMA (optional).
9. Enable the SAD.
10. Enable the DFLT0.

Figure 374 shows a simplified timing diagram when the SAD works with DATCAP[1:0] = 01.

Thanks to the kernel clock (adf_ker_ck), the SAD continuously monitors the audio signal provided by the DFLT0. The threshold is also continuously updated according to the ambient noise level estimation.

- When the SAD detects a sound higher than the programmed threshold (1), the ADF requests the bus clock (adf_bus_ckreq asserted).
- When the bus clock is available (see 2 in Figure 374) then:
  - The data transfer to the memory is triggered.
  - The event interrupt (adf_evt_it) can be generated.
- In this example, the event interrupt (adf_evt_it) is used to wake up the application. The interrupt line is released by clearing SDDETF by writing 1 to it.
- As long as the SAD remains in DETECT state, the application waits to get enough samples and calls, for example the keyword recognition algorithm (see 3 in Figure 374).
- In the case shown in the figure below, the SAD state (SADST) goes back to MONITOR before the keyword is recognized. If DETCFG is set to 1, an event signals when the SAD goes back to MONITOR state. The SAD stops the transfer of samples into the
memory and the application can clean up the receive buffer for the next detection (see 4 in Figure 374).

**Figure 321. SAD timing diagram example**

---

### 30.4.11 Data transfer to memory

#### Data format

The samples processed by DFLT0 are stored into a RXFIFO. The application can read the samples stored into these FIFOs via the ADF digital filter data register 0 (ADF_DFLT0DR). The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The ADF provides 24-bit left-aligned data. Performing a 16-bit access to ADF_DFLT0DR allows the application to get the 16 most significant bits. Performing a 32-bit access to ADF_DFLT0DR allows the application to get a 24-bit data size.

**Figure 322. ADF_DFLTxDR data format**
**Data re-synchronization**

The samples stored into the RXFIFO can be transferred into the memory by using either DMA requests or interrupt signaling.

*Note:* The RXFIFO is located into the adf_ker_ck clock domain, while ADF_DFLT0DR is located into the adf_hclk (AHB) clock domain.

When the AHB clock is available, if ADF_DFLT0DR is empty and if a sample is available into the RXFIFO, this sample is transferred into ADF_DFLT0DR.

The sample transfer from the RXFIFO to ADF_DFLT0DR takes two periods of the AHB clock (adf_hclk) and two periods of the adf_ker_ck clock. The ADF inserts automatically wait-states if the application performs a read operation of ADF_DFLT0DR while the transfer of the new sample from the RXFIFO to ADF_DFLT0DR is not yet completed.

**Data transfer**

The content of the RXFIFO can be transferred to the memory either by using a DMA channel or interrupt services.

Both single and burst, DMA transfers are supported by the ADF, but the application has to care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data re-synchronization explained in the previous section.

*Note:* The burst mode is not available in all products (see the DMA section to check if the product supports it).

In addition, the application can select the RXFIFO threshold (FTH bit) in order to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth/2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the RXFIFO is not empty.

The DMA mode of the RXFIFO is enabled via the DMAEN bit in ADF_DFLT0DR.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as ADF_DFLT0DR is read.
- If FTH = 1, as soon as the RXFIFO reaches the threshold level and a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as one data is read. FTHF is set again if the threshold condition is
met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE/2 data.

**RXFIFO overrun**

A RXFIFO overrun condition is detected when the RXFIFO is full, and a new sample from the DFLT0 must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

*Figure 377* shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO, and the ADF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the adf_hclk line, the gray boxes indicate that the ADF requested the AHB clock. The figure below shows the AHB clock available only when the ADF requests it. In real applications, the AHB clock may also be present if the ADF does not request it.

*Figure 324. Example of overflow and transfer to memory*
30.4.12 Autonomous mode

The ADF can work even if the AHB bus clock is not available (Stop modes). The ADF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the ADF receives a kernel clock if the following conditions are met:
- The ADF autonomous mode is enabled in the RCC.
- The selected kernel clock source is taken from an oscillator available in Stop mode.

In Stop mode, the ADF receives the AHB clock if the following conditions are met:
- The ADF autonomous mode is enabled in the RCC.
- The ADF requests the AHB clock in the following situations:
  - when the ADF must transfer data into memory via the DMA
    The data is directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the ADF needs to generate an interrupt
    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated. The AHB clock request is maintained as long as an enabled interrupt flag is still active.

30.4.13 Register protection

The ADF embeds some hardware protection to prevent invalid situations. The table below shows the list of write-protected and unprotected fields.

<table>
<thead>
<tr>
<th>Registers</th>
<th>Unprotected fields</th>
<th>Write-protected fields</th>
<th>Write-protection condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF global control register (ADF_GCR)</td>
<td>TRGO</td>
<td></td>
<td>DFLTACTIVE0 = 1</td>
</tr>
<tr>
<td>ADF clock generator control register (ADF_CKGR)</td>
<td>CKGDEN, CCK0EN,</td>
<td>PROCDIV[6:0], CCKDIV[3:0],</td>
<td>CKGACTIVE = 1</td>
</tr>
<tr>
<td></td>
<td>CCK1EN</td>
<td>CKGMOD, TRGSRC[3:0],</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRGSENS, CCK[1:0]DIR</td>
<td></td>
</tr>
<tr>
<td>ADF serial interface control register 0 (ADF_SITF0CR)</td>
<td>SITFEN</td>
<td>STH[4:0], SITFMOD[1:0],</td>
<td>SITFACTIVE0X = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCKSRC[1:0]</td>
<td></td>
</tr>
<tr>
<td>ADF bitstream matrix control register 0 (ADF_BSMX0CR)</td>
<td>-</td>
<td>BSSEL[4:0]</td>
<td>DFLTACTIVE0X = 1</td>
</tr>
<tr>
<td>ADF digital filter control register 0 (ADF_DFLT0CR)</td>
<td>DFLTEN</td>
<td>NBDIS[7:0], TRGSRC[3:0],</td>
<td>DFLTACTIVE0X = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRGSENS, FTH, DMAEN,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SNPSFMT, ACQMOD[2:0]</td>
<td></td>
</tr>
<tr>
<td>ADF digital filter configuration register 0 (ADF_DFLT0CF)</td>
<td>SCALE[5:0]</td>
<td>MCICD[8:0], CICMOD[2:0],</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DATSRC[1:0]</td>
<td></td>
</tr>
<tr>
<td>ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)</td>
<td>-</td>
<td>All fields</td>
<td>SKPBF = 1</td>
</tr>
<tr>
<td>ADF delay control register 0 (ADF_DLY0CR)</td>
<td>-</td>
<td>SKPDL[6:0]</td>
<td></td>
</tr>
</tbody>
</table>
All the ADF processing is performed in the adf_proc_ck clock domain. For that reason, enabling or disabling an ADF sub-block may take some time due to the re-synchronization between the AHB clock domain and the adf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected bitfield, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for details.

### 30.5 ADF low-power modes

**Table 268. Effect of low-power modes on ADF**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. ADF interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The ADF registers content is kept. If the autonomous mode is enabled in the RCC and the ADF is clocked by an internal oscillator available in Stop mode, the ADF remains active. The DMA requests are functional and the interrupts in these modes cause the device to exit Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The ADF is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 40.3: ADF implementation for details about Stop modes supported by the ADF.

### 30.6 ADF interrupts

To increase the CPU performance, the ADF offers an interrupt line (adf_flt0_it), sensitive to several events.

*Note:* The status flags are available even if the corresponding interrupt enable flag is not enabled.
The interrupt interface is controlled via the **ADF DFLT0 interrupt enable register (ADF_DFLT0IER)** and the **ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)**.

**Figure 325. ADF interrupt interface**

The table below shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

**Table 269. ADF interrupt requests**

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event/interrupt clearing method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop modes(1)</th>
<th>Exit Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF_FLT0(2)</td>
<td>RXFIFO threshold reached</td>
<td>FTHF</td>
<td>Read ADF_DFLT0DR until RXFIFO level is lower than the threshold.</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>RXFIFO overrun</td>
<td>DOVRF</td>
<td>Write DOVRF to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RSFLT overrun</td>
<td>RFOVRF</td>
<td>Write RFOVRF to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Saturation detection</td>
<td>SATF</td>
<td>Write SATF to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Channel clock absence detection</td>
<td>CKABF</td>
<td>Write CKABF to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAD: sound detected</td>
<td>SDDETF</td>
<td>Write SDDETF to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SAD: sound level value available</td>
<td>SDLVLF</td>
<td>Write SDLVLF to 1.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(1) Only present if the SAD is implemented, refer to section ADF implementation for details.
30.7 ADF application information

30.7.1 ADF configuration examples for audio capture

Table 398 gives some examples of the ADF settings for the digital microphones, focusing on 16 and 48 kHz output data rate.

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also ensure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (adf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple, and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 270. Examples of ADF settings for microphone capture

<table>
<thead>
<tr>
<th>Configuration</th>
<th>ADF setup</th>
<th>Total dec. ratio</th>
<th>FRF</th>
<th>FaDF_CKx (kHz)</th>
<th>FCM (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>1 2 4 64 0x2D (-8.5 dB)</td>
<td>1 x</td>
<td>1.024</td>
<td>64</td>
<td>0.512</td>
</tr>
<tr>
<td>#2</td>
<td>1 2 5 32 0x2B (-14.5 dB)</td>
<td></td>
<td>2.048</td>
<td>64</td>
<td>1.024</td>
</tr>
<tr>
<td>#3</td>
<td>1 2 5 16 0x01 (+3.5 dB)</td>
<td>0 0</td>
<td>3.072</td>
<td>64</td>
<td>0.512</td>
</tr>
<tr>
<td>#4</td>
<td>1 2 5 16 0x01 (+3.5 dB)</td>
<td>0 0</td>
<td>4.096</td>
<td>128</td>
<td>2.048</td>
</tr>
<tr>
<td>#5</td>
<td>1 6 5 8 0x0B (+33.6 dB)</td>
<td>0 0</td>
<td>5.144</td>
<td>64</td>
<td>1.024</td>
</tr>
<tr>
<td>#6</td>
<td>2 2 5 12 0x06 (+18.1 dB)</td>
<td>0 0</td>
<td>6.144</td>
<td>64</td>
<td>2.048</td>
</tr>
<tr>
<td>#7</td>
<td>1 2 5 24 0x2C (-12 dB)</td>
<td>0 0</td>
<td>7.680</td>
<td>80</td>
<td>3.072</td>
</tr>
<tr>
<td>#8</td>
<td>1 2 5 32 0x27 (-26.6 dB)</td>
<td>0 0</td>
<td>9.680</td>
<td>80</td>
<td>3.072</td>
</tr>
<tr>
<td>#9</td>
<td>3 2 5 16 0x02 (+6.0 dB)</td>
<td>0 0</td>
<td>11.520</td>
<td>64</td>
<td>1.024</td>
</tr>
<tr>
<td>#10</td>
<td>2 2 5 24 0x2C (-12 dB)</td>
<td>0 0</td>
<td>12.800</td>
<td>64</td>
<td>1.024</td>
</tr>
<tr>
<td>#11</td>
<td>1 2 5 16 0x01 (+3.5 dB)</td>
<td>0 0</td>
<td>13.680</td>
<td>64</td>
<td>1.024</td>
</tr>
</tbody>
</table>

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.
30.7.2 Programming examples

Example 1

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using only the CIC4, with a decimation of 48, assuming that the kernel clock is 1.536 MHz. Typically, this configuration can be used to detect sound using the SAD.

<table>
<thead>
<tr>
<th>Table 271. Programming sequence (CIC4)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Operations</strong></td>
</tr>
<tr>
<td>Adjust the proper kernel clock frequency via the RCC</td>
</tr>
<tr>
<td>Select the proper ADF kernel clock source via the RCC</td>
</tr>
<tr>
<td>Enable the ADF clocks via the RCC</td>
</tr>
<tr>
<td>Reset the ADF via the RCC</td>
</tr>
<tr>
<td>AFMux programming</td>
</tr>
<tr>
<td>Enable ADF processing clock: ADF_CKGCR = 0x0001 0023</td>
</tr>
<tr>
<td>Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01</td>
</tr>
<tr>
<td>Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000</td>
</tr>
<tr>
<td>Filters configuration (CIC): ADF_DFLT0CICR = 0x0040 2F40</td>
</tr>
<tr>
<td>Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0301</td>
</tr>
<tr>
<td>Micro delay adjust: ADF_DLY0CR = 0x0000 0000</td>
</tr>
<tr>
<td>Enable interrupt events: ADF_DFLT0IER = 0x0000 1000</td>
</tr>
</tbody>
</table>
Example 2

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using the CIC5, and the RSFLT, with a total decimation of 64.

Table 272. Programming sequence (CIC5)

<table>
<thead>
<tr>
<th>Operations</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjust the proper kernel clock frequency via the RCC</td>
<td>Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 6.144 MHz</td>
</tr>
<tr>
<td>Select the proper ADF kernel clock source via the RCC</td>
<td>Refer to the RCC of the product.</td>
</tr>
<tr>
<td>Enable the ADF clocks via the RCC</td>
<td>Refer to the RCC of the product.</td>
</tr>
<tr>
<td>Reset the ADF via the RCC</td>
<td>Refer to the RCC of the product.</td>
</tr>
<tr>
<td>AFMUX programming</td>
<td>Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.</td>
</tr>
<tr>
<td>Enable ADF processing clock: ADF_CKGCIR = 0x0201 0023</td>
<td>PROCDIV = 2 (division by 3): adf_proc_ck frequency is 6.144 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 1.024 MHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.</td>
</tr>
</tbody>
</table>
30.7.3 Connection examples

*Figure 379* shows simple connection examples of the ADF to external sensors.

- **Picture on the left:** two digital microphones connected to the ADF
  
  In this connection, the amount of connections is optimized; DMIC1 and DMIC2 are sharing the same data line, and the same clock line. BSMX allows the application to connect the digital filter either to DMIC1 or to DMIC2. In this configuration when one of the microphone is used, the other is activated as well, as they share the same clock.

- **Picture in the center:** two digital microphones connected to the ADF
  
  In this connection, DMIC1 and DMIC2 are sharing the same data line, but have a dedicated clock line. BSMX allows the application to connect the filter either to DMIC1 or to DMIC2. When the application wants to use a microphone, it is possible to keep the other in low-power mode by forcing its clock line to 0 (CCKyEN = 0, CCKyDIR = 1).
Picture on the right: single sensor connected to the ADF

It is also possible to configure the CCK0 and CCK1 pins to input in order to connect sensors providing the clock.

**Figure 326. Sensor connection examples**

### 30.7.4 Global frequency response

*Figure 380* shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:
- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 40 Hz

The figure below shows the theoretical frequency response using a CIC4 and a CIC5.

*Figure 327. Global frequency response*

*Figure 381* shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:
- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 dB for CIC4.

The -3 dB cut-off frequency is 7061 Hz.
30.7.5 Total ADF gain

This section details how to compute the signal level provided by the ADF according to the filter settings.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a −6 dBFS level (6 dB below full scale).

For example, for the ADF offering a final data width of 24 bits, a signal having an amplitude of $2 \times 10^6$ LSB has a level of:

$$20 \times \log_{10} \left( \frac{2 \times 10^6}{2^{24-1}} \right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left( \frac{\ln(Amp)}{\ln(2)} + 1 \right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of $2 \times 10^6$ LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain ($G_{\text{CIC}}$ and $G_{\text{dBCIC}}$) can be deduced from the following formula giving data size in bits ($DS_{\text{CIC}}$).

$$DS_{\text{CIC}} = (N \times \log_2(D1)) + DS_{\text{in}}$$

where $N$ represents the CIC order (selected by CICMOD[2:0]), and $D1$ is the decimation ratio (given by MCICD[8:0]).

$DS_{\text{in}}$ represents the data size (in bits) of the signal at CIC input.
Warning: DS\textsubscript{CIC} is very important for CIC filter. In order to work fine, DS\textsubscript{CIC} must not exceed 26 bits.

The CIC gain \( G\textsubscript{CIC} \) is given by:

\[ G\textsubscript{CIC} = (D1)^N \]

which gives in decibels:

\[ GdB\textsubscript{CIC} = 20 \times \log_{10}((D1)^N) \]

**Data size at SCALE output**

The data size at SCALE output (including the CIC gain), is a key information as the RSFLT starts to have some saturations, if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted.

The signal amplitude at SCALE output is:

\[ Asout\textsubscript{SCALE} = D1^N \times \frac{GdB\textsubscript{SCALE}}{20} \times Asin\textsubscript{DFLT} \]

\( GdB\textsubscript{SCALE} \) represents the gain selected by SCALE[5:0], in dB.
\( Asout\textsubscript{SCALE} \) is the signal amplitude at SCALE output (in LSB), and \( Asin\textsubscript{DFLT} \) is the signal amplitude at CIC input (LSB).

\[ D\textsubscript{SCALE} = \frac{\ln(Asout\textsubscript{SCALE})}{\ln(2)} + 1 \]

The data size at SCALE output (\( D\textsubscript{SCALE} \)) is expressed in bits.

**RSFLT gain**

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about \( \pm 0.41 \) dB must be considered.

\[ G\textsubscript{RSFLT} = \frac{9.5 dB}{20} = 0.298 \text{ typical} \]

*Note:* The HPF filter has a gain of 0 dB.

**SAD gain**

The SAD is using only the 16 MSB on the signal, as a consequence, from the SAD point of view, the truncation from 24 to 16 bits can be seen as an attenuation.
The figure below shows a simplified view of the filter path and gives, for each significant component, the expression of the bit growth and the gain.

**Figure 329. Simplified DFLT view with gain information**

The table below summarizes the final data size for different filter configurations.

**Table 273. Output signal levels**

<table>
<thead>
<tr>
<th>Filter configurations</th>
<th>Final signal amplitude (LSB)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CIC + RSFLT + HPF + SAD</strong></td>
<td>( A_{\text{out}}<em>{\text{RXFIFO}} = D_1 N \times 10^{\frac{G</em>{\text{dBSCALE}}}{20} \times 9.5^{10} \times A_{\text{in}}_{\text{DFLT}}} )</td>
</tr>
<tr>
<td><strong>Samples provided to the RXFIFO:</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DS\text{SCALE} must be lower than 22 bits</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CIC + RSFLT (+ HPF)</strong></td>
<td>( A_{\text{out}}<em>{\text{HPF}} = D_1 N \times 10^{\frac{G</em>{\text{dBSCALE}}}{20} \times 9.5^{10} \times A_{\text{in}}_{\text{DFLT}}} )</td>
</tr>
<tr>
<td><strong>DS\text{SCALE} must be lower than 22 bits</strong></td>
<td></td>
</tr>
<tr>
<td><strong>CIC (+ HPF)</strong></td>
<td>( A_{\text{out}}<em>{\text{HPF}} = D_1 N \times 10^{\frac{G</em>{\text{dBSCALE}}}{20} \times A_{\text{in}}_{\text{DFLT}}} )</td>
</tr>
<tr>
<td><strong>DS\text{SCALE} must be lower than 24 bits</strong></td>
<td></td>
</tr>
</tbody>
</table>

**G_{\text{dB}}_{\text{SAD}} = -48.1 \text{dB}**

and

**G_{\text{SAD}} = 0.003906**

(1) The SAD is not always implemented (see the ADF implementation section for details.)
**Example using the main filter chain**

If the ADF filter is programmed as follows:
- The input signal is coming from a serial interface (DsinRSFLT = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to -12 dB.
- RSFLT enabled, and the decimation by four is enabled.
- HPF is enabled.

Check first the data size at CIC output:

\[ DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits} \]

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a 22-bit max signal to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

\[ As_{in, SCALE} = 24^5 \times 10^{-12} \times 20^{-10} \times 1 = 2.10^6 \]

\[ DS_{SCALE} = \frac{\ln(2.10^6)}{\ln(2)} + 1 = 21.93 \text{ bits} \]

If a higher gain is used, the RSFLT may saturate the output signal for strong input signals.

At the end, the final signal amplitude is:

\[ As_{out, HPF} = 24^5 \times 10^{-12} \times 10^{-9.5} \times 10^{-10} \times 1 = 5.9711 \times 10^6 \]

\[ Ds_{out, HPF} = \frac{\ln(5.9711 \times 10^6)}{\ln(2)} + 1 \approx 23.51 \text{ bits} \]

or:

\[ SDB_{OUT} = 20 \times \log_{10} \left( \frac{2^{23.51}}{2^{24}} \right) = -2.84 \text{ dBFS} \]

**30.7.6 How to compute SAD thresholds**

The SAD does not compute the RMS value of the converted signal, but the average of the absolute values. As a consequence, the estimated level differs from the RMS value of the signal:
- For a sine signal having an RMS value of 1, the SAD computes a level of 0.9.
- For a white or pink noise signal having an RMS value of 1, the SAD computes a level of about 0.8.
Note: *FRSIZE*[2:0] has a big influence on the accuracy of the level estimation: big *FRSIZE*[2:0] values give better results.

**Threshold programming with SADMOD = 01**

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 63 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02.

The sensitivity of this microphone is typically -26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 63 dBSPL produces a digital signal of about:
- 26 dBFS - (94 - 63) = -57 dBFS.

- **SCALE value adjustment**
  For this example, the filter configuration is the following:
  - CIC5 with a decimation by 16
  - RSFLT enabled with a decimation by 4
  - HPF enabled
  A SCALE value of 3.5 dB is recommended for this configuration. As DFTL0 provides samples only used for a sound detection (samples not provided to the application), a bigger gain value can be applied: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 15.6 dB).

- **Input signal amplitude**
  The input signal is -57 dBFS, corresponding to an amplitude
  \[A_{\text{in}} = 10^{-\frac{57}{20}} = 0.00141 \text{ LSB}\.\]

- **Signal level at SAD input**
  The total filter gain for the SAD is:

  \[G_{\text{SAD}} = 16^5 \times 10^{\frac{15.6}{20}} \times 10^{\frac{9.5}{20}} \times \frac{1}{256} = 73.68 \times 10^3 \text{ or 97.3 dB}\]

  The signal amplitude received by the SAD is:

  \[A_{\text{in,SAD}} = 0.00141 \times G_{\text{SAD}} \sim 104 \text{ LSB}\]

  The gain can be increased if the expected amplitude is too small. For the targeted application, 104 LSB is fine.

  If the input signal is expected to be a sine, the sound level for a signal amplitude of 104 LSB is:

  \[\text{SDLVL} = \frac{A_{\text{in,SAD}} \times \sqrt{2}}{2} \times 0.9 \sim 66 \text{ LSB}\]\n
  where 0.9 is the correction factor to apply with respect to the RMS value.
• Program the trigger value
ANMIN and SNTHR must be programmed to trigger the SAD when the input signal level reaches 66 LSB.

For SADMOD[1:0] = 01, the threshold value is given by:

\[
\text{THRH} = \text{ANMIN} \times \frac{\text{GdB}_{\text{SNTHR}}}{20}
\]

where GdB_{SNTHR} represents the decibel value selected by SNTHR[3:0].

When SNTHR[3:0] = 6 dB for example, this formula becomes:

\[
\text{THRH} = 2 \times \text{ANMIN}
\]

So \(\text{ANMIN} = \frac{\text{THRH}}{2} = \frac{66}{2} = 33 \text{ LSB}\).

In Figure 383, the trigger value (THRH in red) is fixed to 66 LSB. The input signal is at -65dBFS during 256 samples, then its value goes to -55 dB for 256 samples, and finally it is reduced to -60 dBFS.

The blue curve is showing the sound level estimation (SDLVL) versus time. Fluctuation on the estimated value can be observed due to windowing effect of FRSIZE samples.

The SAD DETECT state (when green signal is high) is maintained during four additional frames due to hangover function value.

In this example ANSLP = FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 33.

**Figure 330. SAD example working with SADMOD = 01**

**Threshold programming with SADMOD = 1x**

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 57 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02. The sensitivity of this microphone is typically -26 dBFS for an input signal of 94 dBSPL.
An acoustic signal at 57 dBSPL produces a digital signal of about:
- 26 dBFS - (94-63) = - 63 dBFS.

- Adjust SCALE value
  For this example, the filter configuration is the following:
  - CIC4 with a decimation by 48
  - RSFLT bypassed
  - HPF enabled

A SCALE value of 3.5 dB is recommended for this configuration. The samples provided by DFTL0 are only used for a sound detection, without providing the samples to the application, a bigger gain value can be provided: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 24 dB).

- Input signal amplitude
  The input signal is - 63 dBFS, corresponding to an amplitude:
  \[ \text{Asin} = 10^{(-63/20)} = 0.000708 \text{ LSB}. \]
Signal level at SAD input
The total filter gain for the SAD is:

$$G_{\text{SAD}} = 48^4 \times 10^{24} \times 0.003906 \times 328.6 \times 10^3 = 328.6 \times 10^3 \text{ or } 110.3 \text{ dB}$$

The signal amplitude received by the SAD is:

$$A_{\text{in SAD}} = 0.000708 \times G_{\text{SAD}} \sim 232 \text{ LSB}$$

The gain can be increased if the expected amplitude is too small. 
If the input signal is expected to be a sine, the sound level for a signal amplitude of 232 LSB is:

$$\text{SDLVL} = \frac{A_{\text{in SAD}} \times \sqrt{2}}{2} \times 0.9 - 148 \text{ LSB}$$

where 0.9 is the correction factor to apply with respect to the RMS value.

**Note:** ANLVL converges to average of SDLVL values, with a long constant time. So SDLVL ~ ANLVL = 148 LSB for a constant input signal at 57 dBSPL.

Programming trigger value
For SADMOD = 1', the SAD compares the estimated ambient noise multiplied by the gain selected by SNTHR[3:0] to ANMIN[12:0] * 4. 
For simplification, SNTHR[3:0] is set to 1 (6 dB), meaning that ANLVL is multiplied by two. 
The SAD triggers if $2 \times \text{ANLVL} > \text{THRH}$. 
In this mode, 

$$\text{THRH} = 4 \times \text{ANMIN}$$

So the SAD triggers if:

$$\text{ANLVL} > 2 \times \text{ANMIN}$$

So ANMIN = 148 / 2 = 74 LSB

In **Figure 384**, the trigger value (THR in red) is fixed to 148 LSB. The input signal is at -75 dBFS during 512 samples, then its value goes to -62 dB for 11000 samples, and finally it is reduced to -70 dBFS. 
The blue curve shows the sound level estimation (SDLVL) versus time. The black curve shows the ambient noise estimation versus time, increasing or decreasing logarithmically. During the learning phase, it reaches the SDLVL value. 
In this example ANSLP = 6, FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames),
HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 74.

Figure 331. SAD example working with SADMOD = 1x

30.8  ADF registers

All the ADF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

30.8.1  ADF global control register (ADF_GCR)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 0  TRGO: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also used to synchronize several ADF together by controlling the adf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.
1: Write 1 generates a positive pulse on the adf_trgo signal and triggers the acquisition on enabled filter having their ACQMOD[2:0] = 01x and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.
30.8.2 ADF clock generator control register (ADF_CKGCR)

Address offset: 0x004
Reset value: 0x0000 0000

This register is used to control the clock generator. The clock adf_proc_ck must be enabled before enabling other ADF parts.

```
<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>CKGACTIVE</td>
<td>Clock generator active flag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bit is set and cleared by hardware. It is used by the application to</td>
</tr>
<tr>
<td></td>
<td></td>
<td>check if the clock generator is effectively enabled (active) or not. The</td>
</tr>
<tr>
<td></td>
<td></td>
<td>protected fields of this function can only be updated when CKGACTIVE = 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(see Section 40.4.13 for details). The delay between a transition on</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2 periods of adf_proc_ck.</td>
</tr>
<tr>
<td>30-24</td>
<td>PROCDIV[6:0]</td>
<td>Divider to control the serial interface clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bitfield is set and reset by software. It is used to adjust the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>frequency of the clock provided to the SITF.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$F_{\text{adf_ift_ck}} = \frac{F_{\text{adf_ker_ck}}}{(\text{PROCDIV} + 1)}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This bitfield must not be changed if the filter is enabled (DFTEN = 1).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: adf_ker_ck provided to the SITF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: adf_ker_ck / 2 provided to the SITF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2: adf_ker_ck / 3 provided to the SITF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>127: adf_ker_ck / 128 provided to the SITF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: This bitfield can be write-protected (see Section 40.4.13 for details).</td>
</tr>
<tr>
<td>23-20</td>
<td>Reserved</td>
<td>Must be kept at reset value</td>
</tr>
</tbody>
</table>
```
Bits 19:16 **CCKDIV[3:0]**: Divider to control the ADF_CCK clock

This bitfield is set and reset by software. It is used to adjust the frequency of the ADF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the ADF_CCK is given by the following formula:

\[ F_{ADF\_CCK} = \frac{F_{adf\_ker\_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)} \]

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

- 0000: The ADF_CCK clock is adf_proc_ck.
- 0001: The ADF_CCK clock is adf_proc_ck / 2.
- 0010: The ADF_CCK clock is adf_proc_ck / 3.
- ...
- 1111: The ADF_CCK clock is adf_proc_ck / 16.

*Note: This bitfield can be write-protected (see Section 40.4.13 for details).*

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger the corresponding filter. This bitfield is not significant if the CKGMOD = 0.

- 0000: TRGO selected
- 0010: adf_trg1 selected
- Others: Reserved

*Note: This bitfield can be write-protected (see Section 40.4.13 for details).*

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

- 0: A rising edge event triggers the activation of CKGEN dividers.
- 1: A falling edge even triggers the activation of CKGEN dividers.

*Note: When the trigger source is TRGO, the sensitivity is forced to falling edge, thus TRGSENS value is not taken into account. This bit can be write-protected (see Section 40.4.13 for details).*

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: ADF_CCK1 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK1 pin.

- 0: The ADF_CCK1 pin direction is in input.
- 1: The ADF_CCK1 pin direction is in output.

*Note: This bit can be write-protected (see Section 40.4.13 for details).*

Bit 5 **CCK0DIR**: ADF_CCK0 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK0 pin.

- 0: The ADF_CCK0 pin direction is in input.
- 1: The ADF_CCK0 pin direction is in output.

*Note: This bit can be write-protected (see Section 40.4.13 for details).*

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if the filter is enabled (DFTEN = 1).

- 0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.
- 1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

*Note: This bit can be write-protected (see Section 40.4.13 for details).*
Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: ADF_CCK1 clock enable
This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK1 pin.
- 0: Bitstream clock not generated
- 1: Bitstream clock generated on the ADF_CCK1 pin.

Bit 1 **CCK0EN**: ADF_CCK0 clock enable
This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK0 pin.
- 0: Bitstream clock not generated
- 1: Bitstream clock generated on the ADF_CCK0 pin

Bit 0 **CKGDEN**: CKGEN dividers enable
This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.
- 0: CKGEN dividers disabled
- 1: CKGEN dividers enabled

### 30.8.3 ADF serial interface control register 0 (ADF_SITF0CR)

Address offset: 0x080
Reset value: 0x0000 1F00

This register is used to control the serial interface SITF0.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

- **Bit 31 SITFACTIVE**: Serial interface active flag
  - This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when SITFACTIVE is set to 0 (see Section 40.4.13 for details).
  - The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of adf_proc_ck.
  - 0: The serial interface is not active, and can be configured if needed.
  - 1: The serial interface is active and protected fields cannot be configured.

- **Bits 30:13** Reserved, must be kept at reset value.

- **Bits 12:8 STH[4:0]**: Manchester symbol threshold/SPI threshold
  - This bitfield is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (see Manchester mode for details on computation).
  - In addition this bitfield is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.
  - **Note:** This bitfield can be write-protected (see Section 40.4.13 for details).

- **Bits 7:6** Reserved, must be kept at reset value.
Bits 5:4 **SITFMOD[1:0]**: Serial interface type
   This bitfield is set and cleared by software. It is used to define the serial interface type.
   00: LF_MASTER SPI mode
   01: Normal SPI mode
   10: Manchester mode: rising edge = logic 0, falling edge = logic 1
   11: Manchester mode: rising edge = logic 1, falling edge = logic 0
   *Note:* This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source
   This bitfield is set and cleared by software. It is used to select the clock source of the serial interface.
   00: Serial clock source is ADF_CCK0.
   01: Serial clock source is ADF_CCK1.
   others: reserved
   *Note:* This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 0 **SITFEN**: Serial interface enable
   This bit is set and cleared by software. It is used to enable/disable the serial interface.
   0: Serial interface disabled
   1: Serial interface enabled

### 30.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR)

Address offset: 0x084
Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to DFLT0.

<table>
<thead>
<tr>
<th>BSMXACTIVE</th>
<th>BSSEL[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>rw rw rw rw</td>
</tr>
</tbody>
</table>

Bit 31 **BSMXACTIVE**: BSMX active flag
   This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when BSMXACTIVE is set to 0. This BSMXACTIVE flag cannot go to 0 if DFLT0 is enabled.
   0: BSMX is not active and can be configured if needed.
   1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream selection
   This bitfield is set and cleared by software. It is used to select the bitstream to be processed for DFLT0.
   00000: bs0_r provided to DFLT0
   00001: bs0_f provided to DFLT0
   others: reserved
   *Note:* This bitfield can be write-protected (see Section 40.4.13 for details).
30.8.5 ADF digital filter control register 0 (ADF_DFLT0CR)

Address offset: 0x088
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>DFLTACTIVE</td>
<td>DFLT0 active flag</td>
</tr>
<tr>
<td>30</td>
<td>DFLTRUN</td>
<td>DFLT0 run status flag</td>
</tr>
<tr>
<td>29-28</td>
<td>NBDIS[7:0]</td>
<td>Number of samples to be discarded</td>
</tr>
<tr>
<td>27-20</td>
<td>TRGSRC[3:0]</td>
<td>DFLT0 trigger signal selection</td>
</tr>
<tr>
<td>19-16</td>
<td></td>
<td>Reserved, must be kept at reset value</td>
</tr>
<tr>
<td>15-12</td>
<td></td>
<td>Reserved, must be kept at reset value</td>
</tr>
<tr>
<td>11-9</td>
<td></td>
<td>Reserved, must be kept at reset value</td>
</tr>
</tbody>
</table>

Bit 31 DFLTACTIVE: DFLT0 active flag
This bit is set and cleared by hardware. It indicates if DFLT0 is active: can be running or waiting for events.
0: DFLT0 not active (can be re-enabled again, via DFLTEN bit, if needed)
1: DFLT0 active

Bit 30 DFLTRUN: DFLT0 run status flag
This bit is set and cleared by hardware. It indicates if DFLT0 is running or not.
0: DFLT0 not running and ready to accept a new trigger event
1: DFLT0 running

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 NBDIS[7:0]: Number of samples to be discarded
This bitfield is set and cleared by software. It is used to define the number of samples to be discarded every time DFLT0 is re-started.
0: No sample discarded
1: 1 sample discarded
2: 2 samples discarded
...
255: 255 samples discarded

Note: This bitfield can be write-protected (see Section 40.4.13 for details).

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 TRGSRC[3:0]: DFLT0 trigger signal selection
This bitfield is set and cleared by software. It is used to select which external signals trigger DFLT0.
0000: TRGO selected
0010: adf_trgi selected
Others: Reserved

Note: This bitfield can be write-protected (see Section 40.4.13 for details).

Bits 11:9 Reserved, must be kept at reset value.
Bit 8 **TRGSENS**: DFLT0 trigger sensitivity selection
This bitfield is set and cleared by software. It is used to select the trigger sensitivity of the external signals:
- 0: A rising edge event triggers the acquisition.
- 1: A falling edge event triggers the acquisition.

*Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge. This bit can be write-protected (see Section 40.4.13 for details).*

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: DFLT0 trigger mode
This bitfield is set and cleared by software. It is used to select the filter trigger mode:
- 000: Asynchronous continuous acquisition mode
- 001: Asynchronous single-shot acquisition mode
- 010: Synchronous continuous acquisition mode
- 011: Synchronous single-shot acquisition mode
- 100: Window continuous acquisition mode
- Others: Same as 000

*Note: This bitfield can be write-protected (see Section 40.4.13 for details).*

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection
This bit is set and cleared by software. It is used to select the RXFIFO threshold:
- 0: RXFIFO threshold event generated when the RXFIFO is not empty
- 1: RXFIFO threshold event generated when the RXFIFO is half-full

*Note: This bit can be write-protected (see Section 40.4.13 for details).*

Bit 1 **DMAEN**: DMA requests enable
This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory:
- 0: DMA interface for the corresponding digital filter disabled
- 1: DMA interface for the corresponding digital filter enabled

*Note: This bit can be write-protected (see Section 40.4.13 for details).*

Bit 0 **DFLTEN**: DFLT0 enable
This bit is set and cleared by software. It is used to control the start of acquisition of the DFLT0 path. This bit behavior depends on ACQMOD[2:0] and external events. The serial or parallel interface delivering the samples must be enabled as well:
- 0: Acquisition immediately stopped
- 1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x.
30.8.6 ADF digital filter configuration register 0 (ADF_DFLT0CICR)

Address offset: 0x08C
Reset value: 0x0000 0000

This register is used to control the main CIC filter.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>30-26</td>
<td><strong>SCALE[5:0]</strong></td>
<td>Scaling factor selection</td>
</tr>
<tr>
<td>25-20</td>
<td><strong>MCICD[8:0]</strong></td>
<td>CIC decimation ratio selection</td>
</tr>
<tr>
<td>19-17</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>16-8</td>
<td><strong>DATSRC[1:0]</strong></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

**SCALE[5:0]**: This bitfield is set and cleared by software. It is used to select the gain to be applied at CIC output (see Table 389 for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this bitfield informs the application on the current gain value.

- 000000: 0 dB
- 000001: + 3.5 dB
- 000010: + 6 dB or shift left by 1 bit
- ...
- 011000: + 72 dB or shift left by 12 bits
- 100000: - 48.2 dB or shift right by 8 bits (default value)
- 100001: - 44.6 dB
- 100010: - 42.1 dB or shift right by 7 bits
- 100011: - 38.6 dB
- ...
- 101110: -6 dB or shift right by 1 bit
- 101111: -2.5 dB
- Others: Reserved

**MCICD[8:0]**: This bitfield is set and cleared by software. It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).

- 0: Decimation ratio is 2.
- 1: Decimation ratio is 2.
- 2: Decimation ratio is 3.
- 3: Decimation ratio is 4.
- ...
- 511: Decimation ratio is 512.

Note: This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 7 Reserved, must be kept at reset value.
Bits 6:4 **CICMOD[2:0]**: Select the CIC order

This bitfield is set and cleared by software. It is used to select the order of the MCIC.

- 100: MCIC configured in single Sinc^4 filter
- 101: MCIC configured in single Sinc^5 filter
- Others: Reserved

*Note: This bitfield can be write-protected (see Section 40.4.13 for details).*

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter

This bitfield is set and cleared by software.

- 00: Stream coming from the BSMX selected
- 10: Stream coming from the ADCITF1 selected
- 11: Stream coming from the ADCITF2 selected

*Note: This bitfield can be write-protected (see Section 40.4.13 for details).*

### 30.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)

**Address offset:** 0x090

**Reset value:** 0x0000 0000

This register is used to control the reshape and HPF filter.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 9:8** **HPFC[1:0]**: High-pass filter cut-off frequency

This bitfield is set and cleared by software. It is used to select the cut-off frequency of the high-pass filter. $F_{PCM}$ represents the sampling frequency at HPF input.

- 00: Cut-off frequency $= 0.000625 \times F_{PCM}$
- 01: Cut-off frequency $= 0.00125 \times F_{PCM}$
- 10: Cut-off frequency $= 0.00250 \times F_{PCM}$
- 11: Cut-off frequency $= 0.00950 \times F_{PCM}$

*Note: This bitfield can be write-protected (see Section 40.4.13 for details).*

**Bit 7** **HPFBYP**: High-pass filter bypass

This bit is set and cleared by software. It is used to bypass the high-pass filter.

- 0: HPF not bypassed (default value)
- 1: HPF bypassed

*Note: This bit can be write-protected (see Section 40.4.13 for details).*

**Bits 6:5** Reserved, must be kept at reset value.
Bit 4 RSFLTD: Reshaper filter decimation ratio
This bit is set and cleared by software. It is used to select the decimation ratio for the reshape
filter
0: Decimation ratio is 4 (default value).
1: Decimation ratio is 1.
*Note:* This bit can be write-protected (see Section 40.4.13 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 RSFLTBYP: Reshaper filter bypass
This bit is set and cleared by software. It is used to bypass the reshape filter and its
decimation block.
0: Reshape filter not bypassed (default value)
1: Reshape filter bypassed
*Note:* This bit can be write-protected (see Section 40.4.13 for details).

30.8.8 ADF delay control register 0 (ADF_DLY0CR)

Address offset: 0x0A4
Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

<table>
<thead>
<tr>
<th>SKPBF</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
<th>RES</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Bit 31 SKPBF: Skip busy flag
This bit is set and cleared by hardware. It is used to control if the delay sequence is
completed.
0: ADF ready to accept a new value into SKPDLY[6:0]
1: Last valid SKPDLY[6:0] still under precessing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 SKPDLY[6:0]: Delay to apply to a bitstream
This bitfield is set and cleared by software. It defines the number of input samples that are
skipped. Skipping is applied immediately after writing to this bitfield, if SKPBF = 0 and
DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state
machine.
0: No input sample skipped
1: 1 input sample skipped
...
127: 127 input samples skipped
30.8.9  **ADF DFLT0 interrupt enable register (ADF_DFLT0IER)**

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14  Reserved, must be kept at reset value.

- **Bit 13 SDLVLIE**: SAD sound-level value ready enable
  This bit is set and cleared by software.
  0: Sound-level-ready interrupt disabled
  1: Sound-level-ready interrupt enabled

- **Bit 12 SDDETIE**: Sound activity detection interrupt enable
  This bit is set and cleared by software.
  0: Sound-trigger interrupt disabled
  1: Sound-trigger interrupt enabled

- **Bit 11 RFOVRIE**: Reshape filter overrun interrupt enable
  This bit is set and cleared by software.
  0: Reshape filter overrun interrupt disabled
  1: Reshape filter overrun interrupt enabled

- **Bit 10 CKABIE**: Clock absence detection interrupt enable
  This bit is set and cleared by software.
  0: Clock absence interrupt disabled
  1: Clock absence interrupt enabled

- **Bit 9 SATIE**: Saturation detection interrupt enable
  This bit is set and cleared by software.
  0: Saturation interrupt disabled
  1: Saturation interrupt enabled

Bits 8:2  Reserved, must be kept at reset value.

- **Bit 1 DOVRIE**: Data overflow interrupt enable
  This bit is set and cleared by software.
  0: Data overflow interrupt disabled
  1: Data overflow interrupt enabled

- **Bit 0 FTHIE**: RXFIFO threshold interrupt enable
  This bit is set and cleared by software.
  0: RXFIFO threshold interrupt disabled
  1: RXFIFO threshold interrupt enabled
30.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)

Address offset: 0x0B0
Reset value: 0x0000 0000

This register contains the status flags for the digital filter path.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>r</td>
<td>rc_w1</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVL**: Sound level value ready flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that new sound level value is not ready. Write 0 has no effect.
1: Read 1 means that new sound level value is ready. Write 1 clears this flag.

Bit 12 **SDDETF**: Sound activity detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no sound activity is detected. Write 0 has no effect.
1: Read 1 means that sound activity is detected. Write 1 clears this flag.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RXNEF**: RXFIFO not empty flag
This bit is set and cleared by hardware according to the RXFIFO level.
0: RXFIFO empty
1: RXFIFO not empty

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DOVRF**: Data overflow flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no overflow is detected. Write 0 has no effect.
1: Read 1 means that an overflow is detected; Write 1 clears this flag.
Bit 0 **FTHF**: RXFIFO threshold flag
This bit is set by hardware, and cleared by the hardware when the RXFIFO level is lower than
the threshold.
0: RXFIFO threshold not reached
1: RXFIFO threshold reached

### 30.8.11 ADF SAD control register (ADF_SADCR)

Address offset: 0x0B8
Reset value: 0x0000 0000

This register is used for the configuration and the control of the sound activity detection.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>SADACTIVE: SAD Active flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>30:14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13:12</td>
<td>SADMOD[1:0]: SAD working mode</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

Note: This bitfield can be write-protected (see Section 40.4.13 for details).
Bits 10:8  FRSIZE[2:0]: Frame size
  This bitfield is set and cleared by software. It is used to define the size of one frame and also
to define how many samples are taken into account to compute the short-term signal level.
  000: 8 PCM samples used to compute the short-term signal level
  001: 16 PCM samples used to compute the short-term signal level
  010: 32 PCM samples used to compute the short-term signal level
  011: 64 PCM samples used to compute the short-term signal level
  100: 128 PCM samples used to compute the short-term signal level
  101: 256 PCM samples used to compute the short-term signal level
  11x: 512 PCM samples used to compute the short-term signal level
  Note: This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 7  HYSTEN: Hysteresis enable
  This bit is set and cleared by software. It is used to enable/disable the hysteresis function
  (see Table 389 for details). This bit must be kept to 0 when SADMOD[1:0] = 1x.
  0: Hysteresis function disabled. THR_H is always used.
  1: Hysteresis function enabled. THR_H is used for MONITOR to DETECT transition and THR_L
      is used for DETECT to MONITOR transition.
  Note: This bit can be write-protected (see Section 40.4.13 for details).

Bit 6  Reserved, must be kept at reset value.

Bits 5:4  SADST[1:0]: SAD state
  This bitfield is set and cleared by hardware. It indicates the SAD state and is meaningful only
  when SADEN = 1. The SAD state can be:
    - LEARN when the SAD is in learning phase or in SDLVL computation mode
    - MONITOR when the SAD is in monitoring phase
    - DETECT when the SAD detects a sound
  00: SAD in LEARN state
  01: SAD in MONITOR state
  11: SAD in DETECT state

Bit 3  DETCFG: Sound trigger event configuration
  This bit is set and cleared by software. It is used to define if the sddet_evt event is generated
  only when the SAD enters to MONITOR state or when the SAD enters or exits the DETECT state.
  0: sddet_evt generated when SAD enters the MONITOR state
  1: sddet_evt generated when SAD enters or exits the DETECT state
  Note: This bit can be write-protected (see Section 40.4.13 for details).

Bits 2:1  DATCAP[1:0]: Data capture mode
  This bitfield is set and cleared by software. It is used to define in which conditions, the
  samples provided by DLFT0 are stored into the memory.
  00: Samples from DFLT0 not transfered into the memory
  01: Samples from DFLT0 transfered into the memory when SAD is in DETECT state
  1x: Samples from DFLT0 transfered into memory when SAD and DFLT0 are enabled
  Note: This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 0  SADEN: Sound activity detector enable
  This bit is set and cleared by software. It is used to enable/disable the SAD.
  0: SAD disabled and SAD state reset
  1: SAD enabled
30.8.12 ADF SAD configuration register (ADF_SADCFGR)

Address offset: 0x0BC
Reset value: 0x0000 0000

This register is used for the configuration of the sound activity detection.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **ANMIN[12:0]**: Minimum noise level

This bitfield is set and cleared by software. It is used to define the minimum noise level and then the sensitivity. It represents a positive number.

*Note:* This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HGOVR[2:0]**: Hangover time window

This bitfield is set and cleared by software. Once the SAD state is DETECT, this parameter is used to define the amount of time the sound is allowed to remain below the threshold, before switching the SAD to MONITOR state (see FRSIZE bitfield for the description of a frame).

000: SAD back to MONITOR state if sound is below threshold for 4 frames
001: SAD back to MONITOR state if sound is below threshold for 8 frames
010: SAD back to MONITOR state if sound is below threshold for 16 frames
011: SAD back to MONITOR state if sound is below threshold for 32 frames
100: SAD back to MONITOR state if sound is below threshold for 64 frames
101: SAD back to MONITOR state if sound is below threshold for 128 frames
110: SAD back to MONITOR state if sound is below threshold for 256 frames
111: SAD back to MONITOR state if sound is below threshold for 512 frames

*Note:* This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LFRNB[2:0]**: Number of learning frames

This bitfield is set and cleared by software. It is used to define the number of learning frames to perform the first estimate of the noise level.

000: 2 frames used to compute the initial noise level
001: 4 frames used to compute the initial noise level
010: 8 frames used to compute the initial noise level
011: 16 frames used to compute the initial noise level
1xx: 32 frames used to compute the initial noise level

*Note:* This bitfield can be write-protected (see Section 40.4.13 for details).

Bit 7 Reserved, must be kept at reset value.
30.8.13  ADF SAD sound level register (ADF_SADSDLVR)

Address offset: 0x0C0

Reset value: 0x0000 0000

This register contains the short-term sound-level computed by the SAD.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bits 14:0  SDLVL[14:0]: Short term sound level

This bitfield is set by hardware. It contains the latest sound level computed by the SAD. To refresh this value, SDLVLF must be cleared.
30.8.14  ADF SAD ambient noise level register (ADF_SADANLVR)

Address offset: 0x0C4
Reset value: 0x0000 0000

This register contains the ambient noise level computed by the SAD.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bits 14:0  ANLVL[14:0]: Ambient noise level estimation

This bitfield is set by hardware. It contains the latest ambient noise level computed by the SAD. To refresh this bitfield, the SDLVLF flag must be cleared.

30.8.15  ADF digital filter data register 0 (ADF_DFLT0DR)

Address offset: 0x0F0
Reset value: 0x0000 0000

This register is used to read the data processed by the digital filter.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:8  DR[23:0]: Data processed by DFT0

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

30.8.16  ADF register map

Table 274. ADF register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | ADF_GCR      |    |    |    |    | x  | x  | x  | x  | x  | x  | x  | x  | x  | x  | x  | x  | x  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | TRGO |
| Offset  | Register name      | Offset (hex) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|-------------------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x004   | ADF_CKGCR         |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008 - | Reserved          |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x080   | ADF_SITF0CR       |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x084   | ADF_BSMX0CR       |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x088   | ADF_DFLT0CR       |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08C   | ADF_DFLT0CICR     |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x090   | ADF_DFLT0RSFR     |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x094 - | Reserved          |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0A0   | ADF_DLY0CR        |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0A4   | ADF_DFLT0IER      |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0A8   | ADF_DFLT0ISFR     |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0AC   | ADF_DFLT0ISFR     |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0B0   | ADF_DFLT0ISFR     |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                   | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
### Table 274. ADF register map and reset values (continued)

| Offset (hex) | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x0B4        | Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0B8        | ADF_SADCR     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | SADACTIVE     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0BC        | ADF_SADCFCGR  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | ANMIN[12:0]   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0C0        | ADF_SADSDLVR  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | SDLVL[14:0]   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0C4        | ADF_SADANLVR  |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | ANLVL[14:0]   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x0CB - 0x0EC| Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0F0        | ADF_DFLT0DR   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | DR[23:0]      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|              | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

Refer to Section 2.3 for the register boundary addresses.
31  Digital camera interface pixel pipeline (DCMIPP)

31.1  Introduction

The DCMIPP is the pixel pipeline section of a high-resolution camera subsystem: it gets pixels from a parallel interface, and after some processing (such as decimation, cropping) dumps them to the memory.

DCMIPP supports multiple types of external sensors, among others:
- dumb sensors (without internal ISP), which output raw Bayer pixels
- smart sensors (with an internal ISP), which usually output RGB or YUV pixels
- smart sensors with internal compression, which output a bit-stream (such as JPEG, or H264)

The DCMIPP input interface integrates a parallel interface (up to 16 bits at 120 MHz, with internal/external synchronization).

A first common part of the DCMIPP selects the input exclusively from the parallel interface. Data go to dedicated pipeline(s) before they are sent to memory for further processing or display purposes.

### Table 275. Available pipeline

<table>
<thead>
<tr>
<th>ID</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe0</td>
<td>Dump pipe</td>
<td>Used to dump the received data as-is, without any processing (for example, without pixel extraction, no formatting), with an exception: it formats the 10/12/14-bit raw Bayer components mapped unpacked into 16-bit memory words, to ease the job of a downstream application. 2D crop operation and basic decimation can be performed within this pipe.</td>
</tr>
</tbody>
</table>

*Figure 332* shows the DCMIPP main functions, namely the integrated parallel interface, the applicative post-processing pipeline with its crop, downsize, and pixel formatting.

*Figure 332. DCMIPP overview*
Table 276. Glossary

<table>
<thead>
<tr>
<th>Item</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Alpha component, used to define transparency, opaque = 0xFF, for example ARGB.</td>
</tr>
<tr>
<td>BPC</td>
<td>Bits per component (for example, RGB565 has five bits for the red component, hence 5 bpc)</td>
</tr>
<tr>
<td>BPP</td>
<td>Bits per pixel (for example, RGB565 has 16 bits per pixel, hence 16 bpp)</td>
</tr>
<tr>
<td>DCMIPP</td>
<td>Digital camera interface - Pixel pipeline</td>
</tr>
<tr>
<td>Interlaced</td>
<td>Interlaced video: the field with odd lines is transmitted first, followed, after the last odd line, by the field with even lines. The two fields are thus consecutive but exclusive.</td>
</tr>
<tr>
<td>NPU</td>
<td>Neural network processing engine (also known as NN, neural network).</td>
</tr>
<tr>
<td>Planar</td>
<td>Defines in how many sub-buffers a pixel buffer is split into: 1 (coplanar).</td>
</tr>
<tr>
<td>YUV444</td>
<td>Pixel format, YUV color reference, all three components (Y, U, V) given per pixel.</td>
</tr>
<tr>
<td>YUV422</td>
<td>Pixel format, YUV color reference, chroma component (U, V) sub-sampled 1/2 in X</td>
</tr>
<tr>
<td>YUV420</td>
<td>Pixel format, YUV color reference, chroma component (U, V) sub-sampled 1/2 in X and Y.</td>
</tr>
</tbody>
</table>
31.2 DCMIPP main features

- Parallel input interface:
  - Pixel rate: up to 16 bits in parallel, 100 Mpixel/s (typically 1080p30, maximum 2048x2048 on processing pipelines after decimation).
  - Pixel format: RGB565, RGB888, YUV422, raw Bayer/Mono 8/10/12/14, and ByteStream (JPEG)

- Pipeline maximum resolution:
  - Dimensioned for 5 Mpixel (typically 2560 x 2048 or 2688 x 1944) before decimation
  - Maximum pipe resolution: 4094 x 4094

- Maximum frame per second (fps): typical examples are defined with 30 fps, it is possible to reach higher rates (for example 120 fps) by decreasing image resolution to guarantee the maximum pixel rate constraints.

- Flow selection and frame control:
  - Capture mode in Continuous vs. Snapshot mode

- Decimation on Pipe0 (dump pipe)
  - Pure sub-sampling, ratio 1 vs. 1, 2, 4 in X and ratio 1 vs. 1, 2 in Y

- Parallel pipelines for parallel applications:
  - Pipe0 (for data dump) for a direct dump without processing

- Single Pipe0 (for data dump) for a direct dump from the camera to the memory (with only few operations like crop and basic decimation).

- Output pixel format:
  - Pipe0: any data as-is, Y/Rb: 8/10/12/14 statistics, bit streams

- AXI master:
  - Output FIFOs to drain pixel/data to the memory (linked to the memory latency) from the pipe(s)
  - The unique AXI-Master interface implements one client per pipe
  - Double buffer mode capability per pipe

- Overrun detection:
  - Notification in case of frame impacted by pixel congestion (no impact on the following frame if the congestion cause is solved)
31.3 **DCMIPP functional description**

31.3.1 **DCMIPP block diagram**

The block diagram of the DCMIPP is shown in Figure 333.

![Figure 333. DCMIPP block diagram](image)

**Table 277. DCMIPP input/output pads**

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMIPP_Dn (n = 0 to 15)</td>
<td>Input</td>
<td>Bit n of the parallel data bus coming from the camera sensor</td>
</tr>
<tr>
<td>DCMIPP_PIXCLK</td>
<td>Input</td>
<td>Pixel clock sent by the master (parallel camera sensor module)</td>
</tr>
<tr>
<td>DCMIPP_VSYNC</td>
<td>Input</td>
<td>VSYNC signal (vertical synchronization) coming from the camera sensor</td>
</tr>
<tr>
<td>DCMIPP_HSYNC</td>
<td>Input</td>
<td>HSYNC signal (horizontal synchronization) coming from the camera sensor</td>
</tr>
</tbody>
</table>
### Table 278. DCMIPP input/output pins

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dcmipp_pclk</td>
<td>Input</td>
<td>DCMIPP APB clock (APB bus)</td>
</tr>
<tr>
<td>dcmipp_aclk</td>
<td>Input</td>
<td>DCMIPP AXI clock (AXI bus)</td>
</tr>
<tr>
<td>dcmipp_it_global</td>
<td>Output</td>
<td>DCMIPP interrupts (refer to Table 295 for the full list of interrupt sources)</td>
</tr>
<tr>
<td>dcmipp_p0hsync_evt</td>
<td>Output</td>
<td>Pipe0 Hsync event</td>
</tr>
<tr>
<td>dcmipp_p0vsync_evt</td>
<td>Output</td>
<td>Pipe0 Vsync event</td>
</tr>
<tr>
<td>dcmipp_p0frame_evt</td>
<td></td>
<td>Pipe0 frame end event</td>
</tr>
<tr>
<td>dcmipp_p0line_evt</td>
<td></td>
<td>Pipe0 line event</td>
</tr>
</tbody>
</table>

### Table 279. DCMIPP clocks

<table>
<thead>
<tr>
<th>Domain</th>
<th>Clock</th>
<th>Maximum frequency</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel</td>
<td>dcmipp_pxclk</td>
<td>120 MHz</td>
<td>Up to 2x 120 MB/s in 16-bit mode</td>
</tr>
<tr>
<td>AXI bus</td>
<td>dcmipp_aclk</td>
<td>266 MHz</td>
<td>SoC AXI frequency</td>
</tr>
<tr>
<td>APB bus</td>
<td>dcmipp_pclk</td>
<td>133 MHz</td>
<td>SoC APB frequency</td>
</tr>
</tbody>
</table>

### Table 280. DCMIPP resets

<table>
<thead>
<tr>
<th>Domain</th>
<th>Reset type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel + Pipeline</td>
<td>Asynchronous</td>
<td>Resynchronized internally on dcmipp_pxclk</td>
</tr>
<tr>
<td>AXI bus</td>
<td>Synchronous</td>
<td>Resets directly the dcmipp_aclk domain</td>
</tr>
<tr>
<td>APB bus</td>
<td>Synchronous</td>
<td>Resets directly the dcmipp_pclk domain</td>
</tr>
</tbody>
</table>

### 31.3.3 DCMIPP reset and clocks
Clocks and pixel rate limitations

This paragraph lists the DCMIPP limitations and bottlenecks:

- **The target is to sustain an average pixel rate (PixRateAvg):**
  - A 5 Mpixel sensor at 15 fps requires 75 Mpixel/s

- **Vertical and horizontal blanking:**
  - The full system (sensor, interface, pipe, AXI) embeds limited capacity FIFOs. The idle times that the sensor inserts during its blanking periods impact all the downstream elements, including interface, pipe, and AXI. A pixel rate (PixRatePeak) higher than average must be used. The idle time due to the blanking can reach 33% of the frame duration, sometimes more (check sensor specification).
    - PixRatePeak = 1.33 * PixRateAvg

- **Parallel interface clock (dcmipp_pxclk):**
  - On the parallel interface, pixels are received sequentially, in one, two or three cycles. The maximum implemented clock (120 MHz) constrains the peak pixel rate, depending on the ratio of cycles per pixel.
    - PixRatePeak ≤ dcmipp_pxclk / (1, 2 or 3) = 120 / (1, 2 or 3)
    - Clock frequency:
      - > dcmipp_pxclk ≤ dcmipp_aclk
    - Consequences: see Table 281.

<table>
<thead>
<tr>
<th>Table 281. Parallel interface maximum resolution (80 MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pixel</strong></td>
</tr>
<tr>
<td>Format</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>Mono/raw</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>RGB565</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>RGB888</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>YUV422</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>YUV444</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

- Clock setting: dcmipp_pxclk ≥ PixRatePeak * Ratio(1,2,3)
31.3.4  DCMIPP maximum resolution

The limitations related to the maximum resolution supported by the DCMIPP are:

- Maximum width and height across pipes: 4094 x 4094 pixels
  - Registers and counters are limited to 12 bits
- Maximum performance: 266 Mpixel/s
  - Larger pixel rate can be used, but with lower memory latency
  - Maximum pixel rate on each pipe must remain below the pixel clock
- Max line stride: 32767 bytes
  - Registers and counters, to handle 4094 pixels x 32 bits x 2 (provision for interleaving)

31.3.5  DCMIPP minimum requirements for frame structure

DCMIPP imposes the following minimum requirements for the frame architecture:

**Blanking phase**

- Dump pipe
  - The minimal duration of the blanking between two frames must be an equivalent of 666 datawords of 32 bits

**Data image area**

- Vertical image area: minimum one line
- Horizontal image area: minimum two pixels
- The minimum total frame size is fixed to 36 pixels

31.3.6  Description of DCMIPP pixel format support

*Table 282* lists the supported pixel formats in the input interfaces (parallel interface), and in the output Pipe0.

<table>
<thead>
<tr>
<th>Index(1)</th>
<th>Pixel format</th>
<th>BPP</th>
<th>Parallel input (clk/pix)(2)</th>
<th>Pipe0 output (dump)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13(3)</td>
<td>Byte stream (JPEG, compressed video)</td>
<td>8</td>
<td>Y, 1</td>
<td>Y</td>
</tr>
<tr>
<td>14(4)</td>
<td>Other data</td>
<td>8 to 16(4)</td>
<td>Y, 1</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>Raw6</td>
<td>6</td>
<td>(5)</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>Raw7</td>
<td>7</td>
<td>(5)</td>
<td>Y</td>
</tr>
<tr>
<td>5</td>
<td>Raw8</td>
<td>8</td>
<td>Y, 1</td>
<td>Y</td>
</tr>
<tr>
<td>6</td>
<td>Raw10</td>
<td>10</td>
<td>Y, 1</td>
<td>To 16 bpc</td>
</tr>
<tr>
<td>7</td>
<td>Raw12</td>
<td>12</td>
<td>Y, 1</td>
<td>To 16 bpc</td>
</tr>
<tr>
<td>8</td>
<td>Raw14</td>
<td>14</td>
<td>Y, 1</td>
<td>To 16 bpc</td>
</tr>
<tr>
<td>9</td>
<td>Mono8</td>
<td>8</td>
<td>Y, 1</td>
<td>Y</td>
</tr>
<tr>
<td>10</td>
<td>Mono10</td>
<td>10</td>
<td>Y, 1</td>
<td>To 16 bpc</td>
</tr>
<tr>
<td>11</td>
<td>Mono12</td>
<td>12</td>
<td>Y, 1</td>
<td>To 16 bpc</td>
</tr>
<tr>
<td>12</td>
<td>Mono14</td>
<td>14</td>
<td>Y, 1</td>
<td>To 16 bpc</td>
</tr>
</tbody>
</table>

(1) Index in the table corresponds to the pixel format in the input interfaces.
(2) BPP indicates the number of bits per pixel.
(3) JPEG: Joint Photographic Experts Group
(4) Other data: can include 8 to 16 bits
(5) Y indicates a register that can be disabled.
31.4 DCMIPP input and flow control

This section describes the functional elements of the DCMIPP. For each, it details the features, the software configuration, and provides a software configuration example (when needed).

31.4.1 Parallel input interface

The parallel interface is a flexible pixel interface that clocks in a bus of 8- to 16-bit in parallel, with an externally provided clock, and adds vertical and horizontal synchronization, provided either with specific pins (external sync) or flagged via specific data (as in CCIR601).

Interface (all inputs)

- **Clk**: clock up to 120 MHz, internally named dcmipp_pxclk
- **Data**: 16 IO pins, up to 120 Mbps/pin, 8- to 16-bit, pixels potentially serialized
- **HSync, VSync**, but no enable (EN)
Features

- Horizontal and vertical synchronization:
  - External, via the HSync and VSync pins.
  - Internal, via the CCIR601 flagging.
  
Data provided during the horizontal or vertical blanking cannot be captured.

- Input pixel format (see Section 31.3.6)
  - Binary stream like JPG
  - Single cycle data:
    - Raw Bayer and monochrome: 8/10/12/14 bits in parallel
    - YUV422, RGB565:16 bpp on 16 bits in parallel
  - Double-cycle data:
    - YUV422, RGB565:16 bpp on 8 bits in parallel
    - RGB888/YUV444: 24 bpp on 12 bits in parallel
  - Triple-cycle data:
    - RGB888/YUV444: 24 bpp on 8 bits in parallel

- Flexible input:
  - MSB-vs-LSB bits can be swapped to ease PCB placement
  - Cycle0-vs-Cycle1 can be swapped (for 2-cycle long pixels).

Hardware synchronization mode

In this mode two synchronization signals (DCMIPP_HSYNC and DCMIPP_VSYNC) are used.

Depending on the camera module/mode, data can be transmitted during horizontal/vertical synchronization periods. DCMIPP_HSYNC/DCMIPP_VSYNC act as blanking signals, as all data received during DCMIPP_HSYNC/DCMIPP_VSYNC active periods are ignored.

To correctly transfer images in the RAM buffer, data transfer is synchronized with the DCMIPP_HSYNC/DCMIPP_VSYNC signals. When the hardware synchronization mode is selected, and capture is enabled (CPTREQ bit set in DCMIPP_PxFCTCR), data transfer is synchronized with the deassertion of the DCMIPP_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by the IP-Plug to successive buffers or the same/circular buffer.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using embedded 32-bit codes, using the 0x00/0xFF values not used in data anymore. There are four types of codes, all with 0xFF00 00XY format. The embedded synchronization codes are supported only in 8-bit parallel data capture (in the DCMIPP_PRCR register, the EDM[2:0] bits must be programmed to “000”). For other data widths, this mode generates unpredictable results.

Note: Camera modules generate up to eight synchronization codes when in interleaved mode, while the DCMIPP reacts to one single code. As a consequence, an interleaved flow with an embedded synchronization has one every other frame discarded as not detected.
**Mode 2**

Four embedded codes signal the following events:
- Frame start (FS)
- Frame end (FE)
- Line start (LS)
- Line end (LE)

The XY values in the 0xFF00 00XY format of these codes are programmable (see Section 31.10.9). A 0xFF value programmed as a frame end means that all the unused codes are interpreted as valid frame end codes.

In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.

**Mode 1**

An alternative coding is the camera mode 1. This mode is ITU656 compatible.

The codes signal another set of events:
- SAV (active line) - Line start
- EAV (active line) - Line end
- SAV (blanking) - Start of line during interframe blanking period
- EAV (blanking) - End of line during interframe blanking period

This mode can be supported by programming the following codes:
- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. User can therefore select a bit to compare in the embedded code and detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example:
- FS = 0xA5
- Unmask code for FS = 0x10

In this case the frame start code is embedded in bit 4 of the frame start code.

*Note:* **FEC sequence must be sent before the transfer of the first frame, otherwise other codes are not decoded and the first frame can be lost after the DCMIPP has been enabled. After the first FE sequence, the following frame is captured based on FS sequence.**

**Error conditions**

Error conditions can be detected when using the embedding synchronization data modes. Flags PRERRF in DCMIPP_CMSR2 and ERRF in DCMIPP_PRSR are used for this function. An interruption can be triggered based on error detection if bit PRERRIE in
DCMIPP_CMIER is set for the global interrupt (or bit ERRIE in DCMIPP_PRIER for a local interrupt line).

For instance, such kind of wrong sequences or values generate an error:
- FF 00 00 XY sequence detected with XY value different from the one set for FS/LS/FE/LE
- LS = 0xFF or LE = 0xFF
- If an LS or FS sequence follows an LS sequence

**Software configuration**

**Table 283. DCMIPP_PRCR bit function**

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>Enables the parallel interface</td>
<td>Low-power mode when disabled.</td>
</tr>
</tbody>
</table>
| PCKPOL | Defines clock polarity | – 0: Falling edge
– 1: Rising edge |
| ESS | Defines if VSync and HSync synchronizations are provided by sideband (specific hardware IO) or by in-band (specific embedded codes) signals | – 0: Hardware (external) synchronization: using DCMIPP_HSYNC and DCMIPP_VSYNC pins.
– 1: Embedded (internal) synchronization: using codes FSC, FEC, LSC, LEC and masks in registers DCMIPP_PRESCR and DCMIPP_PRESURy. |
| VSPOL, HSPOL | Defines the polarity of the IO DCMIPP_VSYNC and DCMIPP_HSYNC used for, respectively, the vertical and horizontal synchronization | – 0: Active low
– 1: Active high |
| FORMAT | Defines the pixel format used on the IO DCMIPP_Dx | From 0 to 15 |
| EDM | Defines the amount of valid bits received per cycle | 8/10/12/14/16 bits
The duration of a pixel is extracted by the bit-per-pixel (given by FORMAT) and bit-per-cycle (given by EDM) ratios | 1, 2 or 3 cycles |
| SWAPBITS | Swaps the 16 bits of the parallel interface | 15:0 becomes 0:15 |
| SWAPCYCLES | Swaps, when a pixel is received in two cycles, the data from the first cycle with the data from the second cycle | Software must not activate it when a pixel is received in one or three cycles. |

The embedded synchronization codes are configured by means of DCMIPP_PRESCR and DCMIPP_PRESUR registers (see Table 284).

**Table 284. DCMIPP_PRESCR and DCMIPP_PRESUR bit function**

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSC, FEC</td>
<td>Frame-start and frame-end synchronization, when embedded synchronization ESS = 1</td>
</tr>
<tr>
<td>FSU, FEU</td>
<td>Masks for FSC and FEC codes</td>
</tr>
</tbody>
</table>
Sequence of swap operations

The sequence of the SWAP operators on bits, cycles and RB (red vs. blue) is induced by their functionality:

- the swap on bits (MSB vs. LSB on IOs) occurs first, as operating on the IOs
- the swap on cycles occurs afterwards, as it operates on words already sampled by still raw

31.4.2 Frame counter

A frame counter is available for a tag purpose and counts all the frame received on the selected pipe.

The counter is 32-bit, read-only. It is active at least when bit PIPEN of the pipe is enabled (PIPEN = 1).

It provides an (almost) unique frame number, with a loop time of 4.5 years (if 30 fps).

The counter can be incremented at the FrameStart event of the pipe when the camera sensor module is connected with parallel interface.

The frame counter is cleared by setting CFC in the DCMIPP_CMCR register.

Software configuration

The frame counter must be configured before it is used for the pipe. Software action(s) must be performed in the DCMIPP_CMCR register:

- Frame counter runs as soon as the selected pipe is active (PIPEN = 1)

Note: It is possible to have the Frame counter run on a pipe, while having the pipe not flowing any pixel: this is achieved with pipe enable active (PIPEN = 1) to have the frame counter run, but with CPTREQ = 0 to avoid any pixel flow.

31.4.3 Frame control

The module is replicated for the dump pipe. It handles the frame control and the capture of the pipe, grouping data into frames, and capturing them when requested.
Definitions

There are two notions of frame events

- Event used inside the hardware (provided by the input interfaces):
  - FrameStart: when a frame starts at the input of the pipe (at frame control)
    > Parallel interface with VSync fall
  - FrameEnd: when a frame finishes at the input of pipes (at frame control)
    > Parallel interface: VSync rise (End-of-Frame)

- Events exposed to software via DCMIPP events and interrupts:
  - VSYNC
    > When a frame starts, observed at the input of the pipe (at frame control)
    > It indicates that the current frame has started and has sampled its shadow registers, and that the software can start to configure the next frame
    > It is recommended to use that interrupt to trigger the software that reconfigures the DCMIPP.
  - FRAME END
    > When a frame finishes, observed at the output of pipe (when the last pixel has been written on the AXI)
    > It indicates that the current frame has finished writing all its data to memory, and that the application can immediately start using them
    > It is recommended to use that interrupt to trigger the software that forwards the memory buffer to other applications

Features

- Frame delineation: allows to dump data with the required frame granularity.
  - Extracts the FrameStart and FrameEnd event, based on the input Start-of-Frame and End-of-Frame.
    > The FrameStart immediately generates a VSYNC event.
    > The FrameEnd ripples till the end of the pipe, at the pixel packer level, where it generates the FRAME event.
  - Based on the algorithm detailed in Extraction algorithm for FrameStart and FrameEnd.

- Capture mode: can capture a frame snapshot, or capture continuously.
- Capture rate: can capture all frames, or one out of 2, 4, 8. It is only active when the Capture mode is in Continuous mode.

Extraction algorithm for FrameStart and FrameEnd

The extraction of the FrameStart and FrameEnd is straightforward.

Use cases

The Continuous mode is typically used to send to display or to software-analysis a continuous stream of frames.

The Snapshot mode is typically used to dump a high-resolution frame.

The Frame rate mode is typically used for a low-power continuous analysis, where only one frame every eight is dumped and analyzed.
The frame informations are typically used for interlaced video use cases, where the sensors transmit alternatively odd and even fields (top vs. bottom), and where software must be able to retrieve the type of field. Depending on the sensor, it can be retrieved from:

- the LSB of the counted number of lines in that frame (indicates if the number is odd or even)

**Software configuration**

The frame control supports two different capture modes:

- Snapshot mode: the software captures single frames
- Continuous mode: the software captures a continuous sequence.

The timing diagram of these two modes is shown in *Figure 334*.

---

**Figure 334. Snapshot (CPTMODE = 1) and Continuous (CPTMODE = 0) capture modes**

![Timing diagram of snapshot and continuous capture modes](image)

---

**Table 285. DCMIPP_PxFCTCR bit function**

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPTMODE</td>
<td>Capture mode</td>
<td>0: Continuous</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Snapshot</td>
</tr>
<tr>
<td>CPTREQ</td>
<td>Capture request</td>
<td>0: Capture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>requested</td>
</tr>
</tbody>
</table>
Table 285. DCMIPP_PxFCTCR bit function (continued)

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRATE</td>
<td>Frame rate</td>
<td>– 0: Full rate capture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 1: 1/2-rate capture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 2: 1/4-rate capture</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– 3: 1/8-rate capture</td>
</tr>
<tr>
<td>CPTACT(1)</td>
<td>Capture status</td>
<td>– 1: Capture currently active</td>
</tr>
</tbody>
</table>

1. Bit read from the DCMIPP_PxSR register.
Configuration example

The software operations when using the Snapshot mode are the following:
1. Software sets CPTMODE = 1, to use the Snapshot mode.
2. Software sets PIPEN = 1, to let the flow selection send data to PipeN.
3. Software sets CPTREQ = 1, to request the capture of one frame.
4. At the first following VSVC, the HW samples CPTREQ at 1, with the following impact:
   a) The capture effectively starts: pixels flow into the pipe and are dumped in memory.
   b) CPTACT is set to 1, to mention that a capture is currently ongoing, and that it is best to not modify the configuration of the pipe operators, unless shadowed.
   c) CPTREQ is reset to 0, so that only a single frame is dumped.
5. At the following capture complete interrupt
   a) CPTACT is reset to 0, to signal that the capture is over.
   b) The capture is complete, no more pixels are flowing, hence software can restart to update the configuration of non-shadowed registers without any issue, and use the captured pixels in memory. Depending upon the sensor blanking, this period can be short.
6. At the next VSVC, CPTREQ usually is sampled at 0, so that no more frames are captured
   – As soon as CPTREQ is reset to 0 the software can set it again to 1, to request a capture at the following frame. It means that a continuous sequence of frame can be captured by setting again CPTREQ to 1, continuously frame after frame.

The software operations when using the Continuous mode are the following:
1. Software sets CPTMODE = 0, to use the Continuous mode.
2. Software sets PIPEN = 1, to let the flow selection send data to PipeN.
3. Software sets CPTREQ = 1, to request the continuous capture of frames.
4. At the first following VSVC, CPTREQ is sampled at 1, with the following impact:
   – As in Snapshot mode, the capture starts, and CPTACT is set at 1.
   – Unlike the Snapshot mode, CPTREQ is not modified and remains at 1.
5. At the following capture complete interrupt:
   – As in Snapshot mode, CPTACT is reset to 0, capture stops, software can shortly reconfigure the pipes, and software can use the captured pixels.
6. At the next VSVC, and as long as CPTREQ remains at 1, the capture restarts, similarly as described above.
7. To stop capturing later frames, software resets CPTREQ to 0.
   – An ongoing capture continues until completion.
8. At the next VSVC, CPTREQ is sampled at 0, and the next capture does not restart.

31.4.4 Pipe deactivation

It is possible to abort a pipe to stop any frame acquisition and to potentially offer a mean to reprogram it completely (including non-shadowed registers), or to let it fully disabled for some time. The following operating mode has to be considered to correctly stop the pipe:
1. Disable the CPTREQ bit of the DCMIPP_PxFCTCR register of the corresponding pipe
2. Poll CPTACT = 0 status bit in the DCMIPP_CMSR1 register to check if the pipe is no longer active (idle state)
3. Disable PIPE in the DCMIPP_PxFCTCR register.

When disabling the pipe, the last frame is processed completely by respecting the above operating mode.

Note: To disable the parallel interface (PREN = 0 in the DCMIPP_PxFSCR), it is recommended to first disable the pipe considering the above operation mode, before switching off the parallel interface.

31.5 Pipe0 (dump pipe)

31.5.1 Overview

Pipe0 works as a dump pipe: it extracts data from the camera sensor module and dumps them (as-is) to the targeted memory, with some basic decimation and cropping 2D operations in between.

Pipe0 retrieves data from the camera sensor module connected to the DCMIPP through the 16-bit parallel interface.

The frame controller handles mainly the camera acquisition mode (continuous or snapshot, frame rate), see Section 31.4.3.

When the input data from the camera is valid and supposed to be processed by Pipe0, 2D-cropping and decimation operations can be configured by the software. A dump counter combining with a limit amount of data to be set is offered to handle unknown length of data or to avoid the amount of data to be too wide within a frame.

31.5.2 Decimation

The decimation allows to cheaply downsize a frame.

Based on parallel interface camera module:

- Factor 1x, 2x, 4x at data/byte level depending on the configuration of bit field BSM[1:0] of the DCMIPP_P0PPPCR register. The decimation is not working directly at pixel level, but it is based on either byte or 16-bit word granularity (for padded...
raw 10, for instance) to make equivalent decimation to one pixel out of two. Bit OEBs is used to start from the odd or even byte (or data) to capture first.

- The decimation can also be based on entire line, by capturing all the lines, or one line out of two thanks to bit LSM. Here, again, choice is given to the software to reject odd or even lines within the frame by configuring bit OELS.

### 31.5.3 Crop/statistics selection/suppression

The dump pipe (Pipe0) has the capability to handle 2-D crop processing. The vertical cropping is based on line, thanks to HSYNC event (from the physical IOs or the embedded code detection if selected).

The horizontal area on which the crop is applied is based on data (32-bit wide), and not on pixels since this pipe is not directly considering the pixels. It handles the data flow with 32-bit granularity. The CROP functionality is not supported when the pipe is conveying JPEG format. The ENABLE bit must be kept cleared into DCMIPP_P0SCSZR to avoid any unpredictable behavior.

The area to be captured within the frame has to be specified configuring the registers DCMIPP_P0SCSTR and DCMIPP_P0SCSZR. The starting point on the two axis are set accordingly as well as the width in both directions. It is possible to take the data inside or outside this area by means of bit POSNEG in the DCMIPP_P0SCSZR register. Some sensors can send sensor configuration data and statistics (histogram) on the very first and last lines of a frame. By specifying active the area outside of this window selection (POSNEG = 1), only statistic data are extracted by the way, and the software can decide to apply some processing on the data to correct, such as contrast or exposure.

#### Features

- 2D-Crop operation with granularity:
  - Lines in the vertical axis
  - 32-bit data in the horizontal axis
- Possibility to capture data inside or outside the defined window (valid image data or statistic data)
- Cropping with 0 value for HSIZE[11:0] or VSIZE[11:0] has no crop effect to the dimension for which the 0 value has been applied (vertical or horizontal axis).

#### Software configuration

Registers DCMIPP_P0SCSTR and DCMIPP_P0SCSZR have to be configured to set the crop feature into the dump pipe:

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSM[1:0]</td>
<td>To select how much byte/data have to be captured within a line. This feature is available only for a parallel interface camera module.</td>
</tr>
<tr>
<td>OEBs</td>
<td>Allows the user to chose if the filtering starts from the odd or the even byte.</td>
</tr>
<tr>
<td>LSM</td>
<td>Possibility to filter out one line out of two.</td>
</tr>
<tr>
<td>OELS</td>
<td>The software can select if the filtering rejects the odd or even lines.</td>
</tr>
</tbody>
</table>

Table 286. DCMIPP_P0PPCR bit function
1. Configure the cropped horizontal starting point with HSTART[11:0] and the width with HSIZE[11:0] (both with a 32-bit data granularity).
2. Configure the cropped lines, starting at line VSTART[11:0] and with a height of VSIZE[11:0] (both with a line granularity)
3. If any value HSIZE or VSIZE is set to 0, the hardware does not consider crop operation in the vertical or horizontal dimension for which the 0 value has been applied, so that all the pixels in that dimension are sampled.
4. Select the inner or the outer part of the window for the data capture by configuring POSNEG bit.
5. Enable the crop feature setting ENABLE bit. This bit must be cleared when JPEG is selected as the input format for Pipe0.

Note: Cropping out the picture size (with too large HSTART or VSTART) leads to a not guaranteed processed frame.

31.5.4 Dump counter

The dump counter is present on dump pipe. It is used to count the amount of data that are dumped in that frame, and to potentially limit the amount dumped if too large. It allows the software to know the size of dumped buffer, and to make sure that no dump is made out of a preallocated buffer.

It is specifically useful when dumping a content whose length is unknown prior to reception, like an encoded JPG stream. When dumping a pixel frame, the size is known thanks to the configured width and height of the frame.

The counter is counting 32-bit data for almost all the input formats, even if the value is expressed in number of bytes. The counter increment is 4-bytes granularity for all formats except the JPEG byte stream input mode, for which the counter granularity is 8-bit because the application does not know the amount of dumped data.

Features

- Dump counter, counting the amount of bytes dumped. It counts the amount of dumped 32-bit words (the increment is 4), except when connected to a parallel interface camera and dumping a 8-bit data in byte stream mode (see FORMAT bit field in the DCMIPP_PRCR register), where it counts per byte (the increment is 1).
  - The counter saturates at 0x03FF_FFFF.
  - At FrameEnd, the counter value is recopied in a readable shadow register and the counter itself is reset to start the count of the next frame.
- Dump limit, that clamps the dumped words below a maximum limit:
  - If the counter reaches the limit, all following bytes are deleted until the next VSync.
  - The counter continues to count over the limit, to indicate how many bytes have been received and must have been dumped if not deleted.
- Interrupt limit is triggered when some bytes are deleted due to dump limit crossed.
- Monochrome or raw Bayer padding (alignment of 10/12/14 bpp to 16 bpp) is also taken into account in data volumes.
Software configuration

Table 287. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLE</td>
<td>1: Enables the limit check.</td>
</tr>
<tr>
<td>CNT</td>
<td>Read-only, counts the amount of data with a 4-byte granularity for most of the input formats or with an 8-bit data granularity in JPEG byte stream mode selected into DCMIPP_PRCR FORMAT. The counter increment is based on dumped data at the pipe output standpoint. The value is expressed in bytes, whatever the input format.</td>
</tr>
<tr>
<td>LIMIT</td>
<td>Gives the maximum amount of 32-bit data that can be dumped. Limit value 0 is inconsistent and the processed frame is not guaranteed if ENABLE bit is set.</td>
</tr>
<tr>
<td>LIMITIE</td>
<td>DCMIPP_PxIER.LIMITIE bit enables the interrupt.</td>
</tr>
</tbody>
</table>

Note: Raw Bayer and monochrome pixels on 10/12/14 bpc are padded onto 16 bits, and are thus slightly larger than their 10/12/14 bit size.

31.5.5 Double buffer mode

The dump pipe uses an AXI master interface to dump the data from the internal FIFO to the external memory. In the application, it is possible to handle frame data swapping memory area frame by frame. The double buffer mode fills up this function. There are two memory address registers set to initialize the base addresses of these memories areas. Each start of captured frame event swaps the memory base address to handle double buffering mode.

The double buffer mode can be used to allow post-processing on a buffer (frame buffer) while the other buffer is read to be displayed (display buffer).

Software configuration

Double buffering mode requires an activation, as well as addresses configuration, to define the two memory areas in which data are consecutively stored frame by frame at the output of the pipeline:

- DCMIPP_PxPPM0AR1 and DCMIPP_PxPPM0AR2 must be filled with the memory addresses associated to the double buffer memory locations.
- Bit DBM in the DCMIPP_PxPPCR register(s) is set to enable the double buffer mode.

31.5.6 Pixel packing

This module works on the dump pipe (Pipe0), setting the arriving pixels in memory words.

Features

- DBM addresses: double buffer mode with their associated address locations if the application need to swap memory address for one frame to the next one.
- Pixel format: all usual formats, as indicated in Section 31.3.6.
- Padding: MSB vs. LSB padding of 10/12/14 bit onto 16 bit.
- Multi-line event: event generated (for DMA and INT) at each multiple of sent lines.
31.5.7 Overrun detection

This logic handles flow-control hazards: the DCMIPP is provided a continuous flow of data (via parallel interface) without any wait/hold capability, while downstream of DCMIPP, the access to memory can be temporarily stuck.

In some cases the internal FIFOs of the DCMIPP get full and pixels may be deleted. The overrun detection logic handles such hazard at best.

Features

- Software gets warned in case of an hazard:
  - Interrupt OVR is triggered at the first deletion of a pixel, repeated at every frame, as long as the overrun hazard is present.
  - It is recommended to skip / trash the impacted frame:
    > Overflowing pixels are deleted, so that when the access to memory resumes, the remaining of the line is discarded (not written out). The normal processing resumes at the start of the next line. Potentially, several full lines can get discarded.
- No Frame-to-Frame impact in case of an hazard:
  - VSync resets what is needed in the DCMIPP, so that any hazard on a previous frame has no impact at all on the pixels dumped in next frames.

Software configuration

- Interrupt DCMIPP_PxSR.OVRF is warned in case of hazard.

### Table 288. DCMIPP_PxPPCR bit function

<table>
<thead>
<tr>
<th>Bit ID</th>
<th>Function</th>
<th>Value</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0A</td>
<td>Base address of pixel buffer in memory, 32-bit wide, aligned on 16 bytes</td>
<td>-</td>
<td>The line alignment of the pixel buffers in memory is assumed on 16 bytes. Therefore the buffer base address (i.e. M0A) must be a multiple of 16 bytes.</td>
</tr>
<tr>
<td>FORMAT</td>
<td>Output pixel format</td>
<td>-</td>
<td>– Pipe0: not defined as default input format used for dump. It allows unique formats ByteData, ByteHeader and Mono/raw Bayer 8/10/12/14.</td>
</tr>
<tr>
<td>LINEMULT</td>
<td>Periodicity of the line (HSync) interrupt and event, as power of 2 of LINEMULT configuration (i.e. every 1, 2, …,128 lines).</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
31.6 Pixel format description

This section describes the pixel formats used in the parallel interface and dump pipe.

The support for the pixel formats, per interface (parallel, dump output) are summarized in Table 289.

Note: In the following tables the components are represented with their symbolic color: red and Cr (= V) in red, blue and Cb (= U) in blue, green in green, and Y, monochrome, and all raw Bayer components in gray.

31.6.1 Parallel interface formats

This paragraph describes the input pixel format as supported by the parallel interface of the DCMIPP and the possible swap combinations.

Note: The parallel interface does not input specifically RGB444/RGB555 (and RGB666). However, a sensor with these output can connect them onto the DCMIPP, by selecting RGB565 (and RGB888), and by either connecting the missing bits with the MSB of the sensor output or by strapping them.

Table 289. Parallel interface input pixel formats

<table>
<thead>
<tr>
<th>Index</th>
<th>IO pin</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Byte</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>Raw Bayer 8</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>Raw Bayer 10</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Raw Bayer 12</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Raw Bayer 14</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>Monochrome 8</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>Monochrome 10</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>Monochrome 12</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>Monochrome 14</td>
<td>Cycle 1/1</td>
<td>-</td>
<td>-</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2-1</td>
<td>RGB565</td>
<td>Cycle 1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>8 bits</td>
<td>Cycle 2/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2-2</td>
<td>RGB565</td>
<td>Cycle 1/1</td>
<td>R4</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
<td>G5</td>
<td>G4</td>
<td>G3</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
</tr>
<tr>
<td></td>
<td>16 bits</td>
<td>Cycle 2/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2-2</td>
<td>PCB: Sensor444 to DCMIPP-RGB565</td>
<td>Cycle 1/1</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
<td>R3</td>
<td>G3</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>G3</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>B3</td>
<td>B2</td>
</tr>
<tr>
<td></td>
<td>PCB: Sensor555 to DCMIPP-RGB565</td>
<td>Cycle 1/1</td>
<td>R4</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
<td>G4</td>
<td>G3</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>G4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
</tr>
<tr>
<td>4-1</td>
<td>RGB888</td>
<td>Cycle 1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R7</td>
<td>R6</td>
<td>R5</td>
<td>R4</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
<td>G7</td>
<td>G6</td>
</tr>
<tr>
<td></td>
<td>12 bits</td>
<td>Cycle 2/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>G3</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>B7</td>
<td>B6</td>
<td>B5</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
</tr>
<tr>
<td></td>
<td>YUV444</td>
<td>Cycle 1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>V7</td>
<td>V6</td>
<td>V5</td>
<td>V4</td>
<td>V3</td>
<td>V2</td>
<td>V1</td>
<td>V0</td>
<td>Y7</td>
<td>Y6</td>
<td>Y5</td>
</tr>
<tr>
<td></td>
<td>-12 bits</td>
<td>Cycle 2/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Y3</td>
<td>Y2</td>
<td>Y1</td>
<td>Y0</td>
<td>U7</td>
<td>U6</td>
<td>U5</td>
<td>U4</td>
<td>U3</td>
<td>U2</td>
<td>U1</td>
</tr>
</tbody>
</table>
### Table 289. Parallel interface input pixel formats (continued)

<table>
<thead>
<tr>
<th>Index</th>
<th>IO pin</th>
<th>Cycle 1/3</th>
<th>Cycle 2/3</th>
<th>Cycle 3/3</th>
<th>Cycle 1/3</th>
<th>Cycle 2/3</th>
<th>Cycle 3/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-2</td>
<td></td>
<td>R7 R6 R5 R4 R3 R2 R1 R0</td>
<td>G7 G6 G5 G4 G3 G2 G1 G0</td>
<td>B7 B6 B5 B4 B3 B2 B1 B0</td>
<td>V7 V6 V5 V4 V3 V2 V1 V0</td>
<td>U7 U6 U5 U4 U3 U2 U1 U0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RGB888 - 8 bits</td>
<td>YUV444 - 8 bits</td>
<td>PCB: Sensor 666 to RGB888</td>
<td>YUV422 - 8 bits (YUY'V)</td>
<td>YUV422 - 16 bits (YUY'V)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Cycle 1/3</td>
<td>Cycle 2/3</td>
<td>Cycle 3/3</td>
<td>Cycle 1/3</td>
<td>Cycle 2/3</td>
<td>Cycle 3/3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0</td>
<td>G5 G4 G3 G2 G1 G0 G5 G4</td>
<td>B5 B4 B3 B2 B1 B0 B5 B4</td>
<td>V7 Y7' Y6' Y5' Y4' Y3' Y2' Y1' Y0'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-1</td>
<td></td>
<td>Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0</td>
<td>U7 U6 U5 U4 U3 U2 U1 U0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y7' Y6' Y5' Y4' Y3' Y2' Y1' Y0'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2</td>
<td></td>
<td>Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0</td>
<td>U7 U6 U5 U4 U3 U2 U1 U0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 290. Correspondence between index and DCMIPP_PRCR register values

<table>
<thead>
<tr>
<th>Index</th>
<th>FORMAT[7:0]</th>
<th>EDM[2:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>0x1E</td>
<td>0x00</td>
</tr>
<tr>
<td>1-2</td>
<td>0x1E</td>
<td>0x04</td>
</tr>
<tr>
<td>2</td>
<td>0x22</td>
<td>0x04</td>
</tr>
<tr>
<td>3</td>
<td>0x23</td>
<td>0x00</td>
</tr>
<tr>
<td>4-1</td>
<td>0x24</td>
<td>0x02</td>
</tr>
<tr>
<td>4-2</td>
<td>0x24</td>
<td>0x00</td>
</tr>
<tr>
<td>5</td>
<td>0x2A</td>
<td>0x00</td>
</tr>
<tr>
<td>6</td>
<td>0x2B</td>
<td>0x01</td>
</tr>
<tr>
<td>7</td>
<td>0x2C</td>
<td>0x02</td>
</tr>
<tr>
<td>8</td>
<td>0x2D</td>
<td>0x03</td>
</tr>
<tr>
<td>9</td>
<td>0x4A</td>
<td>0x00</td>
</tr>
<tr>
<td>10</td>
<td>0x4B</td>
<td>0x01</td>
</tr>
<tr>
<td>11</td>
<td>0x4C</td>
<td>0x02</td>
</tr>
<tr>
<td>12</td>
<td>0x4D</td>
<td>0x03</td>
</tr>
<tr>
<td>13</td>
<td>Other values</td>
<td>0x00</td>
</tr>
</tbody>
</table>
To adapt to non-standard sensors, the parallel interface is flexible and allows to swap bits, cycles, components, namely:

- Bits: LSB-vs.-MSB bits are swapped, across the whole 16-bit IO pins.
- Cycles: cycle 1/2 and cycle 2/2 are swapped. No impact for 1 or 3 cycles.

*Table 291* shows these permutations, based on the RGB565 format. It lists the sampled bits of cycles 1 and 2, on pins 0 to 7, and for each of them returns the assignation of the RGB output components.

### 31.6.2 Dump pipe formats

The dump pipe, thanks to its capability to dump, supports:

- Monochrome and raw Bayer pixels on 10/12/14 bits, padded on 16 bpp.

*Table 292* details the support for these additional pixel formats. Other formats are assumed to be pixels with the same width.

#### Table 291. Parallel interface input pixel formats

<table>
<thead>
<tr>
<th>IO pin</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NATIVE RGB565 8 bits</td>
<td>Cycle 1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R4</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
<td>G5</td>
<td>G4</td>
</tr>
<tr>
<td></td>
<td>Cycle 2/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
</tr>
<tr>
<td>SWAP LSB-MSB RGB565 - 8 bits</td>
<td>Cycle 1/2</td>
<td>G3</td>
<td>G4</td>
<td>G5</td>
<td>R0</td>
<td>R1</td>
<td>R2</td>
<td>R3</td>
<td>R4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Cycle 2/2</td>
<td>B0</td>
<td>B1</td>
<td>B2</td>
<td>B3</td>
<td>B4</td>
<td>G0</td>
<td>G1</td>
<td>G2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SWAP cycles RGB565 8 bits</td>
<td>Cycle 1/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>R4</td>
<td>R3</td>
<td>R2</td>
<td>R1</td>
<td>R0</td>
<td>G5</td>
<td>G4</td>
</tr>
<tr>
<td></td>
<td>Cycle 2/2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>G2</td>
<td>G1</td>
<td>G0</td>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B0</td>
</tr>
</tbody>
</table>

#### Table 292. Dump pipe OUTPUT pixel formats(1)

<table>
<thead>
<tr>
<th>Bits</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte/mono/raw 8 bpp</td>
<td>PN + 3</td>
<td>PN + 2</td>
<td>PN + 1</td>
<td>PN + 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAD = 0 Mono/raw 10 bpp</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>G</td>
<td>N</td>
</tr>
<tr>
<td>PAD = 0 Mono/raw 12 bpp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PAD = 0 Mono/raw 14 bpp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>PAD = 1 Mono/raw 10 bpp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>PAD = 1 Mono/raw 12 bpp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>
1. PN indicates Pixel N.

### 31.6.3 **AXI IP-Plug**

The AXI IP-Plug dumps the data of the pipe to the memory via the AXI.

**Features**

- Pipe ID: single ID, as a unique pipe dump (Pipe0) is deployed
- FIFOs: sized to handle external SDRAM
  - Pipe0: 200 MB/s, thus 256 bytes
- Burst length: 64 or 128 bytes for all pipes
- Outstanding capability: four, on 128-byte bursts
- Bandwidth limiter (via an outstanding limitation per pipe):
  - avoids a last-line dump to overload the memory bandwidth
- Unique AXI-QOS: unique QOS for the DCMIPP

**Common software configuration**

The AXI IP-Plug has a common configuration in the DCMIPP_IPGR1/2/3/8 registers.
The common registers R1 and R3 are writable only when the IP-Plug is in Idle mode.

- MEMORYPAGE: for efficient handling of DRAM (bursts do not cross this module).
- PSTART: request to freeze the plug, to reconfigure it safely.
- IDLE: status to mention the plug is frozen, to reconfigure it safely.
- IPPID[7:0], ARCHIID[4:0], REVID[4:0], DID[5:0]: version IDs of the IP-Plug.

**Per-client software configuration**

The AXI IP-Plug has a per-client configuration in registers DCMIPP_IPC0R1/2/3.

The client registers R1 and R3 are writable only when the IP-Plug is in Idle mode (see PSTART in common registers above to lock it, or below for an example).

- Client1 = Pipe0

The configurable per-client features are the following:

- OTR: Outstanding transactions per client (Oustd = OTR + 1)
- TRAFFIC[2:0]: Burst size: best select 128 bytes, thus TRAFFIC = 4
- WLRU[3:0]: Proportion of total bandwidth: WLRU = BWi / sum (all BW)
- SVC: System virtual channel. Keep SVC = 0
- DPREGSTART: Local base-address of start of FIFO for this client.
- DPREGEND: Local base-address of end of FIFO (included word) for this client.

**Configuration computations**

**IP-Plug common configuration:**

- PSTART: set to 1, to put IP-Plug idle and configure its registers G1/3, R1/3.
- IDLE: read and wait till = 1, to get it idle.
- MEMORYPAGE: 4 to optimize for memory pages of 256 bytes.

**IP-Plug per-client configuration:**

- OTR (client): Outstanding per client, among a total of four outstandings.
  - The provided value, OTR, is the outstanding amount minus 1.
- DPREG size (client): RAM shared size, 256 bytes
  - DPREGSTART = 0 (or after another client location)
  - DPREGEND = 2 for ((2 - 0) + 1) x 128 B = 256 bytes for this client. The value to be programmed is 1 because the word is included.
It results with the Client1 assigned to Pipe0:

- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfiguration)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (wait before continuing)

- Video statically mapped onto IP-Plug Client1:
  - DCMIPP_IPC1R1.OTR = 4
  - DCMIPP_IPC1R1.TRAFFIC = 4 (default)
  - DCMIPP_IPC1R3.DPREGSTART = 0
  - DCMIPP_IPC1R3.DPREGEND = 4

- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART= 1 (unlocks IP-Plug after configuration)

31.7 **Shadow registers**

A dump pipe must have the capability to be reconfigured without stopping it. Some registers can be written in the middle of a frame without impacting the actual frame acquisition. The values are stored into some shadow registers before being written in their corresponding physical registers, based on a trigger event like described in Table 293.

Each physical register in the register map has its address increased by 0x200 respect to its own shadow register address.

<table>
<thead>
<tr>
<th>Shadow register</th>
<th>Physical register</th>
<th>Trigger event</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMIPP_P0FCTCR</td>
<td>DCMIPP_P0CFCTCR</td>
<td>Vsync boundaries</td>
</tr>
<tr>
<td>DCMIPP_P0SCSTR</td>
<td>DCMIPP_P0SCSTR</td>
<td>Vsync boundaries</td>
</tr>
<tr>
<td>DCMIPP_P0SCSZR</td>
<td>DCMIPP_P0SCSZR</td>
<td>Vsync boundaries</td>
</tr>
<tr>
<td>DCMIPP_P0PPCCR</td>
<td>DCMIPP_P0CPPCCR</td>
<td>Vsync boundaries</td>
</tr>
<tr>
<td>DCMIPP_P0PPM0AR1</td>
<td>DCMIPP_P0CPPxM0AR1</td>
<td>Vsync boundaries</td>
</tr>
<tr>
<td>DCMIPP_P0PPM0AR2</td>
<td>DCMIPP_P0CPPM0AR2</td>
<td>Vsync boundaries</td>
</tr>
</tbody>
</table>

It is mandatory to refresh the shadow registers within a frame to prepare the context to be ready for the next start of frame, to avoid any unpredictable behavior during the acquisition. The software must then react to the following interrupt sources to launch the shadow registers accesses:

- At VSYNC interrupt (FrameStart at start of pipe) an interrupt is generated, and the software can update the shadow registers loaded into the physical registers at the end of the current frame (so active for the next frame). The software has an entire frame period to complete the update.

- At LINE interrupt line selection can be used also to trigger the software to update the physical registers with the new frame context to consider. The application code must ensure that the update takes place before the end of frame event (current frame). If not, the context can be partially considered from the next frame, dealing with inconsistent settings.
In any case, the software must ensure that the shadow registers are updated before the end of the current frame, to be active from the next frame. Updates started and not completed before the end of current frame result in inconsistent frame acquisition for the next frame (mixing old and new configurations).

It is recommended to change non-shadowed registers when the corresponding pipe is disabled and in an idle state (refer to Section 31.4.4: Pipe deactivation).

31.8 DCMIPP low power modes

This section describes the behavior of the DCMIPP versus the modes listed in Table 294.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop</td>
<td>Peripheral content is kept. It is recommended to follow the pipe(s) disabling procedure before entering it.</td>
</tr>
<tr>
<td>Standby</td>
<td>Powered-down. The peripherals must be reinitialized after exiting this mode.</td>
</tr>
<tr>
<td>Sleep</td>
<td>No effect. DCMIPP is still working in this mode, peripheral interrupts cause the device to exit it.</td>
</tr>
</tbody>
</table>
31.9  **DCMIPP interrupts**

This section describes the DCMIPP interrupts, and more globally, the features that are related to real-time, namely:

- Free-running: DCMIPP can run without CPU involvement
- Shadow registers: relieve timing constrains from the software
- Interrupts: source of interrupt
- Event pins: to drive external IPs, like a DMA

31.9.1 **Free-running DCMIPP**

After a camera flow has been configured and established, it is able to run permanently, without any software involvement. The following features are available:

- Permanent pipe activation, for Pipe0: when set, the pipe dumps frame data, for an unlimited amount of frames.
  - DCMIPP_PxFCTCR.CPTMODE = 0
  - DCMIPP_PxSR.CPTACT = 1
  - DCMIPP_PxFCTCR.FRATExx (depending on desired frame subsampling rate)
- Double buffered frames: the DCMIPP can be defined with two BaseAddress pointers for the output buffers:
  - Software can set up the two BaseAddress pointers to let the hardware reacting to the VSYNC event to change from one buffer BaseAddress to the other one, and this alternatively.

31.9.2 **Interrupts**

The DCMIPP handles three interrupts pins, the third being the OR of the first two. A register set is associated to each interrupt to provide the following functionality:

- Unmasked events: reports an event that occurred and has not yet been cleared
- Masks: to mask unwanted events
- Masked events: not readable, any active unmasked bit triggers the interrupt pin
- Clear register: write to clear any active event

The interrupt pins are active when high: they go high when triggered by an unmasked event, and remain high until all the masked events of that interrupt are cleared.
The interrupts lines and registers are:

1. Parallel interface, with the next interrupt events handled:
   - ERR: bad embedded synchronization detected.
     > Unexpected behavior in the parallel interface, usually not triggered if no issues.
     > It is due to an unexpected sensor behavior, or to a transmission error.

2. Pipe0 (dump), with the next interrupt events handled:
   - OVR: data overflow: the memory was too slow vs. received pixels.
     > Unexpected behavior in Pipe0, usually not triggered if no issues.
     > Same recommendation as for above OVR: skip the impacted frame.
   - LIMIT: received volume is larger than the maximum allowed dump volume.
     > Unexpected behavior in Pipe0, usually not triggered if no issues.
     > The transmitted flow is too long. More space must be allocated in memory to store the transmission of a next frame.
   - VSYNC: main interrupt, where most software can sit.
     > Trigger: at VSync (mid blanking), permanent (even if no dump active).
     > Typically to reconfigure slowly the pipes (at least shadow registers) for next frame
     > Typically to trigger usage of the previously captured frame.
   - FRAME: secondary interrupt, as backup for fast-software
     > Trigger: after last data dump of this pipe, inactive if no capture has occurred.
     > Typically to quickly reconfigure non-shadow registers (during only vertical blanking)
     > Typically to trigger usage of the previously captured frame.
   - LINE: interruption for stripe-based operators trigger: after every 1/2/4/8/16/32/64/128 dumped lines and last line. It is measured and extracted at end of pipe, close to output to memory.
     > Typically to trigger fast/reactive software or hardware stripe-based operators.

Note: The LINE flag (LINEF bit) is set at the end of frame, to trigger out software event when the frame height is not a multiple of the selected number of lines for which the LINEF event is triggered. The last part of the frame can, in such case, have a lower number of lines than the selection. The event/interrupt is generated even if the number of lines is incomplete. The software must consider that this last event has a lower than expected number of lines when reaching the end of frame (FRAME event).

3. Common interrupt: groups the events of the above interrupts in a single register set and drives its own interrupt line.
   - The third interrupt is designed for systems where a single driver handles the whole DCMIPP: a single access to this interrupt status allows to retrieve the status of the whole DCMIPP.
   - The first two interrupts are designed for systems where a driver handles a pipe and (potentially) another one drives the parallel interface. In that case, each driver handles its own individual interrupt register set, without need for semaphores to arbitrate conflicting accesses.

There is also a common interrupt that can be generated if enabled, when a transfer error is detected during an AXI transfer from the IP-Plug to memories. Error flag
ATXERR in the DCMIPP_CMSR2 register is set and an interrupt can be generated to inform the software about this error status. There is no specific hardware action linked to this error, it is up to software to handle the transfer error situation.

The DCMIPP interrupts are summarized in Table 295. An event that generates an interrupt does it at two locations:
- once in the common interrupt register set (when driven by a single CPU)
- once in the local interrupt register set (when driven by multiple CPUs).

Table 295 provides the local (register and bit) and global (bit only, as all bits are in the same DCMIPP_CMSR2 register) locations.

### Table 295. DCMIPP interrupts

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Local register</th>
<th>Local bit</th>
<th>Common bit in DCMIPP_CMSR2</th>
<th>Event flag/interrupt clearing method</th>
<th>Interrupt enable control bit local/common register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronization error in parallel interface</td>
<td>DCMIPP_PRSR</td>
<td>ERRF</td>
<td>PRERRF</td>
<td>Write CERRF = 1 / CPRERRF = 1</td>
<td>ERRIE/PRERRFIE</td>
</tr>
<tr>
<td>AXI transfer error</td>
<td>-</td>
<td>-</td>
<td>ATXERR</td>
<td>Write CATXERRF = 1</td>
<td>ATXERRIE</td>
</tr>
<tr>
<td>Overflow in Pipe0</td>
<td>DCMIPP_P0SR</td>
<td>OVRF</td>
<td>P0OVRF</td>
<td>Write COVRF = 1 / CP0OVRF = 1</td>
<td>OVRIE/P0OVRIE</td>
</tr>
<tr>
<td>Limit violation in Pipe0</td>
<td></td>
<td>LIMITF</td>
<td>P0LIMITF</td>
<td>Write CLIMITF = 1 / CP0LIMITF = 1</td>
<td>LIMITIE/P0LIMITIE</td>
</tr>
<tr>
<td>Frame start (VSYNC) in Pipe0</td>
<td>DCMIPP_P0SR</td>
<td>VSYNC</td>
<td>P0VSYNC</td>
<td>Write CVSYNCF = 1 / CP0VSYNCF = 1</td>
<td>VSYNCE/P0VSYNCE</td>
</tr>
<tr>
<td>Frame end (FRAME) in Pipe0</td>
<td></td>
<td>FRAMEF</td>
<td>P0FRAMEF</td>
<td>Write CFRAMEF = 1 / CP0FRAMEF = 1</td>
<td>FRAMEIE/P0FRAMEIE</td>
</tr>
<tr>
<td>Multi-line (LINE) in Pipe0</td>
<td></td>
<td>LINEF</td>
<td>P0LINEF</td>
<td>Write CLINEF = 1 / CP0LINEF = 1</td>
<td>LINEFIE/P0LINEFIE</td>
</tr>
</tbody>
</table>

### 31.9.3 Event pins

Event pins are exposed to let an ancillary HW IP, like a central DMA, to synchronize with them. The event pins are the unmasked synchronization events (VSYNC, FRAME, LINE, HSYNC) of the pipes.

An internal events generates a pulse (15 APB cycles long) on its pin.

### Table 296. Event connection

<table>
<thead>
<tr>
<th>Pipe</th>
<th>Event name</th>
<th>Internal signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe0</td>
<td>HSYNC</td>
<td>dcmipp_p0hsync_evt</td>
</tr>
<tr>
<td></td>
<td>VSYNC</td>
<td>dcmipp_p0vsync_evt</td>
</tr>
<tr>
<td></td>
<td>Frame</td>
<td>dcmipp_p0frame_evt</td>
</tr>
<tr>
<td></td>
<td>Line</td>
<td>dcmipp_p0line_evt</td>
</tr>
</tbody>
</table>
31.10 **DCMIPP registers**

The registers are split into groups with a same prefix, as shown in *Table 297*.

<table>
<thead>
<tr>
<th>Register name</th>
<th>Offset</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCMIPP_IP</td>
<td>0x000</td>
<td>IP-Plug registers</td>
</tr>
<tr>
<td>DCMIPP_PR</td>
<td>0x100</td>
<td>Parallel interface</td>
</tr>
<tr>
<td>DCMIPP_CM</td>
<td>0x200</td>
<td>Common registers</td>
</tr>
<tr>
<td>DCMIPP_P0</td>
<td>0x500</td>
<td>Pipe0 (dump pipe) with shadow registers</td>
</tr>
<tr>
<td></td>
<td>0x700</td>
<td>Pipe0 (dump pipe) with physical registers</td>
</tr>
<tr>
<td>DCMIPP_</td>
<td>0xFF0</td>
<td>IP version and configuration (without prefix)</td>
</tr>
</tbody>
</table>

### 31.10.1 DCMIPP IP-Plug global register 1 (DCMIPP_IPGR1)

Address offset: 0x000  
Reset value: 0x0000 0002

| Bit 31:25 Reserved, must be kept at reset value. |
| Bit 24 **QOS_MODE**: Quality of service  
Set of functions enabling to build and configure an architecture meeting bandwidth and latency requirements. |
| Bit 23:3 Reserved, must be kept at reset value. |
| Bit 2:0 **MEMORYPAGE[2:0]**: Memory page size, as power of 2 of 64-byte units:  
0x0: 64 bytes  
0x1: 128 bytes  
0x2: 256 bytes  
0x3: 512 bytes  
0x4: 1K bytes  
0x5: 2K bytes  
0x6: 4K bytes  
0x7: 8K bytes |
### 31.10.2 DCMIPP IP-Plug global register 2 (DCMIPP_IPGR2)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>PSTART: Request to lock the IP-Plug, to allow reconfiguration.</td>
<td></td>
</tr>
</tbody>
</table>
0: No lock requested, IP-Plug runs on demand by background HW.  
1: Lock requested: IP-Plug freezes shortly (see IDLE bit when lock is active).  
PSTART must be reset to 0 after configuration is completed, to restart the IP-Plug. |

### 31.10.3 DCMIPP IP-Plug global register 3 (DCMIPP_IPGR3)

Address offset: 0x008  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>IDLE: Status of IP-Plug</td>
<td></td>
</tr>
</tbody>
</table>
0: IP-Plug is running (on demand by background HW)  
1: IP-Plug is currently locked and can be reconfigured  
IDLE is set after a request by setting PSTART at 1, and reset by resetting PSTART at 0. |

### 31.10.4 DCMIPP IP-Plug identification register (DCMIPP_IPGR8)

Address offset: 0x01C  
Reset value: 0xAA04 0314

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:1</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>DID[5:0]: 5-bit Unique Device Identifier for this instance of the DCMIPP IP-Plug.</td>
<td></td>
</tr>
</tbody>
</table>
The combination of ARCHIID[4:0] and DID[5:0] identifies the specific IP-Plug instance in the family. |
31.10.5 DCMIPP IP-Plug Clientx register 1 (DCMIPP_IPCxR1)

Address offset: 0x020 + 0x10 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 9:8 OTR[1:0]: Maximum outstanding transactions
   0: Disabled. No outstanding transaction limitation (except via FIFO size)
   1: Maximum two outstanding transactions ongoing.
   ... 3: Maximum four outstanding transactions ongoing.
   Other values are not allowed.

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

Bits 2:0 TRAFFIC[2:0]: Burst size as power of 2 of 8-byte units
   0x0: 8 bytes
   0x1: 16 bytes
   0x2: 32 bytes
   0x3: 64 bytes
   0x4: 128 bytes
   Other values: Reserved
31.10.6  DCMIPP IP-Plug Clientx register 2 (DCMIPP_IPCxR2)

Address offset: 0x024 + 0x10 * (x - 1), (x = 1 to 1)
Reset value: 0x0001 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WLRU[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

A client gets a portion of the total bandwidth = Ratio(client) / Sum(all ratios)
0x0: Ratio part = 1
0x1: Ratio part = 2
...
0xFF: Ratio part = 16

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

Bits 11:8  Reserved, must be kept at reset value.

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

31.10.7  DCMIPP IP-Plug Clientx register 3 (DCMIPP_IPCxR3)

Address offset: 0x028 + 0x10 * (x - 1), (x = 1 to 1)
Reset value: 0x001F 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPREGEND[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:21  Reserved, must be kept at reset value.

Bits 20:16  DPREGEND[4:0]: End word (AXI width = 64 bits) of the FIFO of Clientx.
The addressed word is included in the FIFO, so that next DPREGSTART is DPREGEND + 1.

Bits 15:5  Reserved, must be kept at reset value.

Bits 4:0  DPREGSTART[4:0]: Start word (AXI width = 64 bits) of the FIFO of Clientx.
31.10.8 DCMIPP parallel interface control register (DCMIPP_PRCR)

Address offset: 0x104
Reset value: 0x0000 0000

![Table](image)

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

Bit 26 **SWAPBITS**: Swap LSB vs. MSB within each received component
- 0: As received
- 1: Swapped MSB vs. LSB

Bit 25 **SWAPCYCLES**: Swap data (cycle 0 vs. cycle 1) for pixels received on two cycles
- 0: Default
- 1: Swap active: the data of cycle 1 is used before the data of cycle 0.
  The swap must not be activated by software for pixels received in one or three cycles.

Bit 24 Reserved, must be kept at reset value.

Bits 23:16 **FORMAT[7:0]**:
- 0x1E: YUV422
- 0x22: RGB565
- 0x24: RGB888 (= YUV444)
- 0x2A: RAW8
- 0x2B: RAW10
- 0x2C: RAW12
- 0x2D: RAW14
- 0x4A: monochrome 8-bit
- 0x4B: monochrome 10-bit
- 0x4C: monochrome 12-bit
- 0x4D: monochrome 14-bit
- 0x5A: byte stream (JPEG, compressed video)
  Other values: data are captured and output as-is only through the data/dump pipeline
  (for example JPEG or byte input format).
  The monochrome Y input is inserted in the pipe as YUV pixels, with the U and V components
  set to neutral, to represent a gray color.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE**: Parallel interface enable
- 0: Parallel interface disabled to lower power consumption
- 1: Parallel interface enabled
  The parallel interface configuration registers must be correctly programmed before enabling
  this bit.

Bit 13 Reserved, must be kept at reset value.
Bits 12:10 **EDM[2:0]**: Extended data mode
- 0x0: Interface captures 8-bit data on every pixel clock
- 0x1: Interface captures 10-bit data on every pixel clock
- 0x2: Interface captures 12-bit data on every pixel clock
- 0x3: Interface captures 14-bit data on every pixel clock
- 0x4: Interface captures 16-bit data on every pixel clock
Other values: Reserved

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **VSPOL**: Vertical synchronization polarity
- This bit indicates the level on the VSYNC pin when the data are not valid on the parallel interface.
  - 0: VSYNC active low
  - 1: VSYNC active high

Bit 6 **HSPOL**: Horizontal synchronization polarity
- This bit indicates the level on the HSYNC pin when the data are not valid on the parallel interface.
  - 0: HSYNC active low
  - 1: HSYNC active high

Bit 5 **PCKPOL**: Pixel clock polarity
- This bit configures the capture edge of the pixel clock
  - 0: Falling edge active
  - 1: Rising edge active

Bit 4 **ESS**: Embedded synchronization select
- 0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the HSYNC/VSYNC signals.
- 1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
  Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when this bit is set.

Bits 3:0 Reserved, must be kept at reset value.

### 31.10.9 DCMIPP parallel interface embedded synchronization code register (DCMIPP_PRESER)

Address offset: 0x108

Reset value: 0x0000 0000
31.10.10 DCMIPP parallel interface embedded synchronization unmask register (DCMIPP_PRESUR)

Address offset: 0x10C
Reset value: 0x0000 0000

Bits 31:24 **FEC[7:0]**: Frame end delimiter code
This byte specifies the code of the frame end delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, FEC.
If FEC is programmed to 0xFF, all the unused codes (0xFF00 00XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code
This byte specifies the code of the line end delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code
This byte specifies the code of the line start delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code
This byte specifies the code of the frame start delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, FSC.
If FSC is programmed to 0xFF, no frame start delimiter is detected, but the first occurrence of LSC after an FEC code is interpreted as the start of frame delimiter.

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask
This byte specifies the mask to be applied to the code of the frame end delimiter.
0: The corresponding bit in the FEC byte in DCMIPP_ESCR is masked while comparing the frame end delimiter with the received data.
1: The corresponding bit in the FEC byte in DCMIPP_ESCR is compared while comparing the frame end delimiter with the received data.

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask
This byte specifies the mask to be applied to the code of the line end delimiter.
0: The corresponding bit in the LEC byte in DCMIPP_ESCR is masked while comparing the line end delimiter with the received data.
1: The corresponding bit in the LEC byte in DCMIPP_ESCR is compared while comparing the line end delimiter with the received data.

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask
This byte specifies the mask to be applied to the code of the line start delimiter.
0: The corresponding bit in the LSC byte in DCMIPP_ESCR is masked while comparing the line start delimiter with the received data.
1: The corresponding bit in the LSC byte in DCMIPP_ESCR is compared while comparing the line start delimiter with the received data.

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask
This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMIPP_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMIPP_ESCR is compared while comparing the frame start delimiter with the received data.
### 31.10.11 DCMIPP parallel interface interrupt enable register (DCMIPP_PRIER)

Address offset: 0x1F4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:7 Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>ERRIE: Synchronization error interrupt enable</th>
</tr>
</thead>
</table>
| 0: No interrupt generation  
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order. |
This bit is available only in embedded synchronization mode. |

<table>
<thead>
<tr>
<th>Bits 5:0 Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

### 31.10.12 DCMIPP parallel interface status register (DCMIPP_PRSR)

Address offset: 0x1F8  
Reset value: 0x0003 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:18 Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

**Bits 7:0** **FSU[7:0]:** Frame start delimiter unmask  
This byte specifies the mask to be applied to the code of the frame start delimiter.  
0: The corresponding bit in the FSC byte in DCMIPP_ESCR is masked while comparing the frame start delimiter with the received data  
1: The corresponding bit in the FSC byte in DCMIPP_ESCR is compared while comparing the frame start delimiter with the received data
Bit 17 **VSYNC:**
This bit gives the state of the VSYNC pin with the correct programmed polarity if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in VPOL bit, and cleared otherwise.
When embedded synchronization codes are used:
0: Active frame
1: Synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMIPP_CR is set.

Bit 16 **HSYNC:**
This bit gives the state of the HSYNC pin with the correct programmed polarity if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in HPOL bit, and cleared otherwise.
When embedded synchronization codes are used:
0: Active line
1: Synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMIPP_CR is set.

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

Bit 6 **ERRF:** Synchronization error raw interrupt status
0: No synchronization error detected
1: Embedded synchronization characters are not received in the correct order.
This bit is valid only in the embedded synchronization mode. It is cleared by writing 1 to the CERRF bit in DCMIPP_PRFCR.
This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

### 31.10.13 DCMIPP parallel interface interrupt clear register (DCMIPP_PRFCR)

Address offset: 0x1FC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CERRF:** Synchronization error interrupt status clear
Writing a 1 into this bit clears the ERRF bit in DCMIPP_PRSR.
This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

### 31.10.14 DCMIPP common configuration register (DCMIPP_CMCR)
Address offset: 0x204
Reset value: 0x0000 0000

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

Bit 4 **CFC**: Clear frame counter
When this bit is set, the frame counter associated to a pipe is cleared. It resets DCMIPP_CMFRCR register. This bit is always read at 0.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 Reserved, must be kept at reset value.

### 31.10.15 DCMIPP common frame counter register (DCMIPP_CMFRCR)
Address offset: 0x208
Reset value: 0x0000 0000

Bits 31:0 **FRMCNT[31:0]**: Frame counter, read-only, loops around.
Incremented following VSYNC detection mapped to the pipe configured into bits PSFC[1:0] of the DCMIPP_CMCR register. The counter is cleared using the CRC bit of that register.

### 31.10.16 DCMIPP common interrupt enable register (DCMIPP_CMIER)
Address offset: 0x3F0
Reset value: 0x0000 0000

Bits 31:0 **P0 OVR IE**, **P0 LIMIT IE**, **P0 VSYNC IE**, **P0 FRAME IE**, **P0 LINE IE**, **PR ERR IE**, **ATX ERR IE**: Read-write

Bit 31  Reserved, must be kept at reset value.
Bits 30:28  Reserved, must be kept at reset value.
Bit 27  Reserved, must be kept at reset value.
Bits 26:24  Reserved, must be kept at reset value.
Bit 23  Reserved, must be kept at reset value.
Bits 22:20  Reserved, must be kept at reset value.
Bit 19  Reserved, must be kept at reset value.
Bits 18:16  Reserved, must be kept at reset value.
Bit 15 **P0OVRIE**: Overrun interrupt enable for Pipe0
  0: No interrupt generation
  1: An interrupt is generated

Bit 14 **P0LIMITIE**: Limit interrupt enable for Pipe0
  0: No interrupt generation
  1: An interrupt is generated

Bits 13:11  Reserved, must be kept at reset value.
Bit 10  **P0VSYNCIE**: Vertical sync interrupt enable for Pipe0
  0: No interrupt generation
  1: An interrupt is generated

Bit 9  **P0FRAMEIE**: Frame capture complete interrupt enable for Pipe0
  0: No interrupt generation
  1: An interrupt is generated

Bit 8  **P0LINEIE**: Multi-line capture complete interrupt enable for Pipe0
  0: No interrupt generation
  1: An interrupt is generated

Bit 7  Reserved, must be kept at reset value.
Bit 6  **PRERRIE**: Limit interrupt enable for the parallel Interface
  0: No interrupt generation
  1: An interrupt is generated

Bit 5  **ATXERRIE**: AXI transfer error interrupt enable for IP-Plug
  0: No interrupt generation
  1: An interrupt is generated

Bits 4:0  Reserved, must be kept at reset value.

### 31.10.17  DCMIPP common status register 1 (DCMIPP_CMSR1)

Address offset: 0x3F4
Reset value: 0x0000 0003
Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 P0CPTACT: Active frame capture (active from start-of-frame to frame complete) for Pipe0

0: No capture currently active
1: Capture currently active

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:2 Reserved, must be kept at reset value.

Bit 1 PRVSYNC:
This bit gives the state of the VSYNC pin with the correct programmed polarity on the parallel interface if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in VPOL bit of the DCMIPP_PRCR register, and cleared otherwise.

When embedded synchronization codes are used, the meaning of this bit is the following:
0: Active frame
1: Synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in the DCMIPP_PRCR register is set.

Bit 0 PRHSYNC:
This bit gives the state of the HSYNC pin with the correct programmed polarity on the parallel interface if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in HPOL bit of the DCMIPP_PRCR register, and cleared otherwise.

When embedded synchronization codes are used the meaning of this bit is the following:
0: Active line
1: Synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in the DCMIPP_PRCR register is set.

### 31.10.18 DCMIPP common status register 2 (DCMIPP_CMSR2)

Address offset: 0x3F8

Reset value: 0x0000 0000
### Digital camera interface pixel pipeline (DCMIPP)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Bits 31:28
Reserved, must be kept at reset value.

#### Bit 27
Reserved, must be kept at reset value.

#### Bits 26:24
Reserved, must be kept at reset value.

#### Bit 23
Reserved, must be kept at reset value.

#### Bits 22:20
Reserved, must be kept at reset value.

#### Bit 19
Reserved, must be kept at reset value.

#### Bits 18:16
Reserved, must be kept at reset value.

#### Bit 15 **P0OVRF**: Overrun raw interrupt status for Pipe0
- **0**: No data buffer overrun occurred
- **1**: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the CP0OVRF bit in the DCMIPP_CMFCR register.

#### Bit 14 **P0LIMITF**: Limit raw interrupt status for Pipe0
This bit is set when the data counter DCMIPP_P0DCCNT reaches its maximum value DCMIPP_P0DCLIMIT. It is cleared by writing 1 to the CP0LIMITF bit in the DCMIPP_CMFCR register.

#### Bits 13:11
Reserved, must be kept at reset value.

#### Bit 10 **P0VSYNC**: VSYNC raw interrupt status for Pipe0
This bit is set when the VSYNC signal changes from the inactive state to the active state. In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR. It is cleared by writing 1 to the CP0VSYNCF bit in the DCMIPP_CMFCR register.

#### Bit 9 **P0FRAMEF**: Frame capture completed raw interrupt status for Pipe0
- **0**: No capture or ongoing capture
- **1**: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured. In case of a cropped window, this bit is set at the end of line of the last line in the crop, even if the captured frame is empty (for example window cropped outside the frame). This bit is cleared by writing 1 to the CP0FRAMEF bit in the DCMIPP_CMFCR register.

#### Bit 8 **P0LINEF**: Multi-line capture completed raw interrupt status for Pipe0
This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P0PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame. In the case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set. It is cleared by writing 1 to the CP0LINEF bit in the DCMIPP_CMFCR register.

#### Bit 7 Reserved, must be kept at reset value.
Bit 6 **PRERRF**: Synchronization error raw interrupt status for the parallel interface.
- 0: No synchronization error detected
- 1: Embedded synchronization characters are not received in the correct order.
  This bit is valid only in the embedded synchronization mode. It is cleared by writing 1 to the CPRERRRF bit in the DCMIPP_CMFCR register.
  This bit is available only in embedded synchronization mode.

Bit 5 **ATXERRF**: AXI transfer error interrupt status flag for the IP-Plug.
- 0: No AXI transfer error detected
- 1: AXI transfer error occurred on an AXI client. This bit signals an error on a client without any specific hardware action, the software must handle the situation (normally used when debugging software application code).
  This bit is cleared by writing 1 to CATXERRRF bit in the DCMIPP_CMFCR register.

Bits 4:0 Reserved, must be kept at reset value.

## 31.10.19 DCMIPP common interrupt clear register (DCMIPP_CMFCR)

Address offset: 0x3FC

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **CP0OVRF**: Overrun interrupt status clear
- Writing 1 into this bit clears the P0OVRF bit in the DCMIPP_CMSR2 register

Bit 14 **CP0LIMITF**: limit interrupt status clear
- Writing 1 into this bit clears P0LIMITF in the DCMIPP_CMSR2 register

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **CP0VSYNCF**: Vertical synchronization interrupt status clear
- Writing 1 into this bit clears the P0VSYNCF bit in the DCMIPP_CMSR2 register.

Bit 9 **CP0FRAMEF**: Frame capture complete interrupt status clear
- Writing 1 into this bit clears the P0FRAMEF bit in the DCMIPP_CMSR2 register.
Bit 8 **CP0LINEF**: Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P0LINEF in the DCMIPP_CMSR2 register.

Bit 7 **Reserved**, must be kept at reset value.

Bit 6 **CPRERRF**: Synchronization error interrupt status clear
Writing 1 into this bit clears the PRERRF bit in the DCMIPP_CMSR2 register.
This bit is available only in embedded synchronization mode.

Bit 5 **CATXERRF**: AXI transfer error interrupt status clear
Writing 1 into this bit clears the ATXERRF bit in the DCMIPP_CMSR2 register.

Bits 4:0 **Reserved**, must be kept at reset value.

### 31.10.20 DCMIPP Pipe0 flow selection configuration register
(DCMIPP_P0FSCR)

Address offset: 0x404

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>PIPEN</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31**: **PIPEN**: Activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

*Note*: This bit is not shadowed, differently from all other bits in this register.

Bits 30:0 **Reserved**, must be kept at reset value.

### 31.10.21 DCMIPP Pipe0 flow control configuration register
(DCMIPP_P0FCTCR)

Address offset: 0x500

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:4**: Reserved, must be kept at reset value.
Bit 3 CPTREQ: Capture requested
- 0: Capture not requested for next frame
- 1: Capture requested for next frame
When PIPEN = 1 and CPTREQ = 1, the pipe waits for the first VSync, automatically starts a capture, and sets CPTACT = 1 to mention it.
In Snapshot mode the CPTREQ bit is cleared at the start of the first received frame.
In Continuous grab mode, the capture remains active and CPTREQ = 1 until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.
The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 CPTMODE: Capture mode
- 0: Continuous grab mode - The received data are transferred into the destination memory through the AXI master.
- 1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 FRATE[1:0]: Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.
- 00: All frames are captured
- 01: one out of two frames captured (50% bandwidth reduction)
- 10: one out of four frames captured (75% bandwidth reduction)
- 11: one out of eight frames captured (87% bandwidth reduction)

31.10.22 DCMIPP Pipe0 statistic/crop start register (DCMIPP_P0SCSTR)
Address offset: 0x504
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.
Bits 27:16 VSTART[11:0]: Vertical start, from 0 to 4094 pixels high
Bits 15:12 Reserved, must be kept at reset value.
Bits 11:0 HSTART[11:0]: Horizontal start, from 0 to 4094 words wide

31.10.23 DCMIPP Pipe0 statistic/crop size register (DCMIPP_P0SCSZR)
Address offset: 0x508
Reset value: 0x0000 0000
Bit 31 **ENABLE**:  
This bit is set and cleared by software.  
0: Bypass. All the data are computed, if the statistic data are sent within the frame, they are sent to the processing pipe as pixels data.  
1: Enable. Depending on bit POSNEG value, the rectangle defined by VSIZE, HSIZE, VSTART and HSTART can be used to extract or to remove some data (statistical extraction or removal, or basic 2D crop features).  
if POSNEG = 0, the data inside the rectangle area are transmitted (it can correspond to a statistical data removal, or as a crop feature in a data valid image area).  
if POSNEG = 1, the data outside of the rectangle area are transmitted (it can correspond to a statistical data extraction, rejecting all data inside the window).  
This bit must be kept cleared if the input format is JPEG, to avoid unpredictable behavior of the pipe.

Bit 30 **POSNEG**:  
This bit is set and cleared by software. It has a meaning only if ENABLE bit is set.  
0: Positive area, the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART  
1: Negative area, the area excluding the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high  
If the value is maintained at 0 when enabling the crop by means of ENABLE bit, the crop operation is not performed on vertical direction.

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

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 word wide (data 32-bit)  
If the value is maintained at 0 when enabling the crop by means of ENABLE bit, the crop operation is not performed on horizontal direction.

### 31.10.24 DCMIPP Pipe0 dump counter register (DCMIPP_P0DCCNTR)

Address offset: 0x5B0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CNT[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.
Bits 25:0 **CNT[25:0]**: Number of data dumped during the frame. The size of the data is expressed in bytes. It counts only the data selected by means of the CROP 2D function. The counter saturates at 0x3FFFFFF. Granularity is 32-bit for all the formats except for the byte stream formats (for example JPEG) having byte granularity.

### 31.10.25 DCMIPP Pipe0 dump limit register (DCMIPP_P0DCLMTR)

Address offset: 0x5B4
Reset value: 0x00FF FFFF

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>ENABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0: Disabled, no check on the amount of 32-bit words transmitted</td>
</tr>
<tr>
<td></td>
<td>1: Enabled, check done versus limit</td>
</tr>
</tbody>
</table>

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:0 **LIMIT[23:0]**: Maximum number of 32-bit data that can be dumped during a frame, after the crop 2D operation.

### 31.10.26 DCMIPP Pipe0 pixel packer configuration register (DCMIPP_P0PPCR)

Address offset: 0x5C0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:17</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 16</td>
<td><strong>DBM:</strong> Double buffer mode</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: No double buffer mode activated. Pipe0 always dumps to memory address set by DCMIPP_P0PPM0AR1.</td>
</tr>
<tr>
<td></td>
<td>1: Double buffer mode activated. Dump address location switches from DCMIPP_P0PPM0AR1 to DCMIPP_P0PPM0AR2 alternatively on each frame.</td>
</tr>
</tbody>
</table>
Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt
- 0x0: Event after one line
- 0x1: Event after two lines
- 0x2: Event after four lines
- 0x3: Event after eight lines
- 0x4: Event after sixteen lines
- 0x5: Event after 32 lines
- 0x6: Event after 64 lines
- 0x7: Event after 128 lines

Bit 12 Reserved, must be kept at reset value.

Bit 11 **OELS**: Odd/even line select (line select start)
This bit works in conjunction with LSM field (LSM = 1).
- 0: Interface captures first line after the frame start, second one is dropped
- 1: Interface captures second line from the frame start, first one is dropped

Bit 10 **LSM**: Line select mode
- 0: Interface captures all received lines
- 1: Interface captures one line out of two

Bit 9 **OEBLS**: Odd/even byte select (byte select start)
This bit works in conjunction with BSM field (BSM ≠ 00)
- 0: Interface captures the first data (byte or double byte) from the frame/line start, the second one is dropped
- 1: Interface captures the second data (byte or double byte) from the frame/line start, the first one is dropped

Bits 8:7 **BSM[1:0]**: Byte select mode
- 00: Interface captures all received data
- 01: Interface captures 1 data out of 2
- 10: Interface captures one byte out of four
- 11: Interface captures two bytes out of four

Modes 10 and 11 work only with EDM [2:0] = 000 into the DCMIPP_PRCR register.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **PAD**: Pad mode for monochrome and raw Bayer 10/12/14 bpp (MSB vs. LSB alignment)
- 0: Aligns on LSB (and pads null bits on MSB), for backward compatibility with former DCMI.
- 1: Aligns on MSB (and pads null bits on LSB), for better ease of software or GPU.

Bits 4:0 Reserved, must be kept at reset value.

**31.10.27 DCMIPP Pipe0 pixel packer Memory0 address register 1**
(DCMIPP_P0PPM0AR1)

Address offset: 0x5C4
Reset value: 0x0000 0000
31.10.28  **DCMIPP Pipe0 pixel packer Memory0 address register 2**  
(DCMIPP_P0PPM0AR2)

Address offset: 0x5C8  
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Bits 31:0  **M0A[31:0]: Memory0 address**  
Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

31.10.29  **DCMIPP Pipe0 interrupt enable register**  
(DCMIPP_P0IER)

Address offset: 0x5F4  
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Bits 31:0  **M0A[31:0]: Memory0 address**  
Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

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

Bit 7  **OVRIE**: Overrun interrupt enable  
0: No interrupt generation  
1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received.
Bit 6 **LIMITIE:** Limit interrupt enable
   0: No interrupt generation when the limit is reached
   1: An interrupt is generated when the limit is reached

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCE:** VSYNC interrupt enable
   0: No interrupt generation
   1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable
   0: No interrupt generation
   1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable
   0: No interrupt generation when the line is received
   1: An interrupt is generated after the full capture of a group of lines (or last line reached)

### 31.10.30 DCMIPP Pipe0 status register (DCMIPP_P0SR)

**Address offset:** 0x5F8  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31:24 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 23 CPTACT:** Capture immediate status
   0: Capture currently inactive
   1: Capture currently active
   This bit is automatically reset at the end of frame capture complete event (after all the data of that frame have been captured and the IP-Plug has started to emit the last burst on the AXI, usually before the next VSync).

**Bits 22:17 Reserved, must be kept at reset value.**

**Bits 16:8 Reserved, must be kept at reset value.**

**Bit 7 OVRF:** Overrun raw interrupt status
   0: No data buffer overrun occurred
   1: A data buffer overrun occurred and this frame data are corrupted
   This bit is cleared by writing 1 to the COVRF bit in the DCMIPP_P0FCR register.

**Bit 6 LIMITF:** Limit raw interrupt status
   This bit is set when the data counter DCMIPP_PxDCNTR reaches its maximum value DCMIPP_PxDCLIMITR.
   It is cleared by writing 1 to the CLIMITF bit in the DCMIPP_P0FCR register.

**Bits 5:3 Reserved, must be kept at reset value.**
31.10.31  DCMIPP Pipe0 interrupt clear register (DCMIPP_P0FCR)

Address offset: 0x5FC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

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

Bit 7  COVRF: Overrun interrupt status clear
Writing 1 into this bit clears the OVRF bit in the DCMIPP_P0SR register.

Bit 6  CLIMITF: Limit interrupt status clear
Writing 1 into this bit clears LIMITF in the DCMIPP_P0SR register.

Bits 5:3  Reserved, must be kept at reset value.

Bit 2  CVSYNCF: Vertical synchronization interrupt status clear
Writing 1 into this bit clears the VSYNC bit in the DCMIPP_P0SR register.

Bit 1  FRAMEF: Frame capture complete interrupt status clear
Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P0SR register.

Bit 0  CLINEF: Multi-line capture complete interrupt status clear
Writing 1 into this bit clears LINEF in the DCMIPP_P0SR register.
31.10.32  DCMIPP Pipe0 current flow control configuration register  
(DCMIPP_P0CFCTCR)

Address offset: 0x700
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:4 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 3 CPTREQ: Capture requested
0: Capture not requested for next frame.
1: Capture requested for next frame.
When PIPEN = 1, and when the CPTREQ is set to 1 the pipe waits for the first VSync, and automatically starts a capture and sets CPTACT = 1 to mention it.
In Snapshot mode the CPTREQ bit is automatically cleared at the start of the first frame received.
In continuous grab mode the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.
The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 CPTMODE: Capture mode
0: Continuous grab mode - The received data are transferred into the destination memory through the AXI master.
1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame the CPTACT bit is automatically reset.

Bits 1:0 FRATE[1:0]: Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.
00: all frames are captured
01: one out of two frames captured (50% bandwidth reduction)
10: one out of four frames captured (75% bandwidth reduction)
11: one out of eight frames captured (87% bandwidth reduction)

31.10.33  DCMIPP Pipe0 current statistic/crop start register  
(DCMIPP_P0CSCSTR)

Address offset: 0x704
Reset value: 0x0000 0000
31.10.34 DCMIPP Pipe0 current statistic/crop size register (DCMIPP_P0CSCSZR)

Address offset: 0x708

Reset value: 0x0000 0000

Bit 31 ENABLE: Current value of the ENABLE bit
0: Bypass. All data are computed, if the statistics data are sent within the frame, they are sent to the processing pipe as pixels data.
1: Enable: Depending on bit POSNEG value, the rectangle defined by the VSIZE, HSIZE, VSTART, HSTART can be used to extract or to remove certain amount of data (statistical extraction or removal, or basic 2D crop features)
if POSNEG = 0, the data inside the rectangle area are transmitted (can correspond to a statistical data removal, or as a crop feature in a data valid image area).
if POSNEG = 1, the data outside of the rectangle area are transmitted (can correspond to a statistical data extraction, rejecting all data inside the window).

Bit 30 POSNEG: Current value of the POSNEG bit
This bit has a meaning only if ENABLE bit is set.
0: Positive area. The rectangle defined by VSIZE, HSIZE, and VSTART, HSTART is the active area.
1: Negative area. The active area is the area excluding the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:16 VSIZE[11:0]: Current vertical size, from 0 to 4094 pixels high.
If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.
31.10.35 DCMIPP Pipe0 current pixel packer configuration register (DCMIPP_P0CPPPCR)

Address offset: 0x7C0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DBM</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>LINEMULT[2:0]</td>
<td>OELS</td>
<td>LSM</td>
<td>OEBs</td>
<td>BSM[1:0]</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td>PAD</td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 16 **DBM**: Double buffer mode
- 0: No double buffer mode activated. Pipe0 is always dump to memory address set by DCMIPP_P0PPM0AR1 register
- 1: Double buffer mode activated. Dump address location switches from DCMIPP_P0PPM0AR1 register to DCMIPP_P0PPM0AR2 register alternatively on each frame.

Bits 15:13 **LINEMULT[2:0]**: Current amount of capture completed lines for LINE event and interrupt
- 0x0: Event after every line
- 0x1: Event after two lines
- 0x2: Event after four lines
- 0x3: Event after eight lines
- 0x4: Event after sixteen lines
- 0x5: Event after 32 lines
- 0x6: Event after 64 lines
- 0x7: Event after 128 lines

Bit 12 Reserved, must be kept at reset value.

Bit 11 **OELS**: Current odd/even line select (line select start)
This bit works in conjunction with LSM field (LSM = 1)
- 0: Interface captures the first line after the frame start, the second one is dropped
- 1: Interface captures the second line from the frame start, the first one is dropped

Bit 10 **LSM**: Current Line select mode
- 0: Interface captures all received lines
- 1: Interface captures one line out of two

Bit 9 **OEBs**: Current odd/even byte select (byte select start)
This bit works in conjunction with BSM field (BSM ≠ 00)
- 0: Interface captures the first data (byte or double byte) from the frame/line start, the second one is dropped
- 1: Interface captures the second data (byte or double byte) from the frame/line start, the first one is dropped
Bits 8:7  **BSM[1:0]**: Current Byte select mode
00: Interface captures all received data
01: Interface captures one data out of two
10: Interface captures one byte out of four
11: Interface captures two bytes out of four
Modes 10 and 11 work only with EDM [2:0] = 000 into the DCMIPP_PRCR register.

Bit 6  Reserved, must be kept at reset value.

Bit 5  **PAD**: Current Pad mode for monochrome and raw Bayer 10/12/14 bpp (MSB vs. LSB alignment)
0: Aligns on LSB (and pads null bits on MSB), for backward compatibility with former DCMI
1: Aligns on MSB (and pads null bits on LSB), for better ease of software or GPU

Bits 4:0  Reserved, must be kept at reset value.

### 31.10.36 DCMIPP Pipe0 current pixel packer Memory0 address register 1
**(DCMIPP_P0CPPM0AR1)**

Address offset: 0x7C4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **M0A[31:0]**: Memory0 address
Base address of the current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3.0 are always at 0x0.

### 31.10.37 DCMIPP Pipe0 current pixel packer Memory0 address register 2
**(DCMIPP_P0CPPM0AR2)**

Address offset: 0x7C8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **M0A[31:0]**: Memory0 address
Base address of the current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3.0 are always at 0x0.
## 31.11 DCMIPP register map

**Table 298. DCMIPP register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DCMIPP_IPGR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>DCMIPP_IPGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>DCMIPP_IPGR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>DCMIPP_IPGR8</td>
<td>IPP[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>DCMIPP_IPC1R1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>DCMIPP_IPC1R2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WLRU[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>DCMIPP_IPC1R3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DPREGEND[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DPREGSTART[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x104</td>
<td>DCMIPP_PCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x108</td>
<td>DCMIPP_PSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x10C</td>
<td>DCMIPP_PSCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x110</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x114</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x118</td>
<td>DCMIPP_PRI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x11C</td>
<td>DCMIPP_PRS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x120</td>
<td>DCMIPP_PRSC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1358/3791 RM0477 Rev 6
<p>| Offset | Register name     | 31| 30| 29| 28| 27| 26| 25| 24| 23| 22| 21| 20| 19| 18| 17| 16| 15| 14| 13| 12| 11| 10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x200  | Reserved         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x204  | DCMIPP_CMCR      |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x208  | DCMIPP_CMFRCR    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x20C- | 0x3EC            |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x3F0  | DCMIPP_CMIER     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x3F4  | DCMIPP_CMISR1    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x3F8  | DCMIPP_CMISR2    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x3FC  | DCMIPP_CMFCR     |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x400  | Reserved         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x404  | DCMIPP_P0FSCR    |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x408- | 0x4FC            |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x500  | DCMIPP_P0FCTCR   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x504  | DCMIPP_P0SCSTR   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x508  | DCMIPP_P0SCSZR   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x50C- | 0x5AC            |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x5B0  | DCMIPP_P0DCCTR   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x5B4  | DCMIPP_P0DLMTR   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 0x5B8  | Reserved         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5C0</td>
<td>DCMIPP_P0PPPCR</td>
<td>0x5C4</td>
<td>DCMIPP_P0PPM0AR1</td>
<td>0x5C8</td>
<td>DCMIPP_P0PPM0AR2</td>
<td>0x5CC</td>
<td>Reserved</td>
<td>0x5F0</td>
<td>Reserved</td>
<td>0x5F4</td>
<td>DCMIPP_P0IER</td>
<td>0x5F8</td>
<td>DCMIPP_P0SR</td>
<td>0x5FC</td>
<td>DCMIPP_P0FCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x700</td>
<td>DCMIPP_P0CFCTCR</td>
<td>0x704</td>
<td>DCMIPP_P0CSCSTR</td>
<td>0x708</td>
<td>DCMIPP_P0CSCSZR</td>
<td>0x70C</td>
<td>Reserved</td>
<td>0x710</td>
<td>DCMIPP_P0CPPPCR</td>
<td>0x714</td>
<td>DCMIPP_P0CPPM0AR1</td>
<td>0x718</td>
<td>DCMIPP_P0CPPM0AR2</td>
</tr>
</tbody>
</table>

Table 298. DCMIPP register map and reset values (continued)
32 Parallel synchronous slave interface (PSSI)

32.1 Introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

32.2 PSSI main features

The PSSI peripheral main features are the following:
- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) and Ready (PSSI_RDY) alternate function
  When selected, these signals can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.
- Clock out mode

32.3 PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

*Figure 336* shows the PSSI block diagram.
32.3.1 PSSI block diagram

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. *Table 299* shows the PSSI pins.

### Table 299. PSSI input/output pins

<table>
<thead>
<tr>
<th>PSSI signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSSI_PDCK</td>
<td>Input/output</td>
<td>Parallel data clock input/output</td>
</tr>
<tr>
<td>PSSI_D[15:0]</td>
<td>Input/output</td>
<td>Data output when transmitting, data input when receiving</td>
</tr>
</tbody>
</table>

32.3.2 PSSI pins and internal signals

*Figure 336. PSSI block diagram*

*Figure 337. Top-level block diagram*
32.3.3 **PSSI clock**

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

The PSSI_PDCK can be configured in input mode (default) or in output mode. When configured in output mode, the clock is generated by the device RCC.

32.3.4 **PSSI data management**

**Data direction**

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.
Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:
- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly perform transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:28) corresponds to the forth byte transferred. Table 301 illustrates the positioning of the data bytes in two 32-bit words.

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:24</th>
<th>23:16</th>
<th>15:8</th>
<th>7:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D_{n+3}[7:0]</td>
<td>D_{n+2}[7:0]</td>
<td>D_{n+1}[7:0]</td>
<td>D_{n}[7:0]</td>
</tr>
<tr>
<td>4</td>
<td>D_{n+7}[7:0]</td>
<td>D_{n+6}[7:0]</td>
<td>D_{n+5}[7:0]</td>
<td>D_{n+4}[7:0]</td>
</tr>
</tbody>
</table>
16-bit data

When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. Table 302 illustrates the positioning of the data in two 32-bit words.

<table>
<thead>
<tr>
<th>Byte address</th>
<th>31:16</th>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>D_{n+1}[15:0]</td>
<td>D_{n}[15:0]</td>
</tr>
<tr>
<td>4</td>
<td>D_{n+3}[15:0]</td>
<td>D_{n+2}[15:0]</td>
</tr>
</tbody>
</table>

FIFO data buffer and error conditions

An eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

32.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input

The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is set to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.
If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

**Ready (PSSI_RDY) alternate function output**

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples
the data (D[15:0]). If the FIFO has enough free space to receive more data, the PSSI drives the PSSI_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the PSSI drives the PSSI_RDY signal inactive. The transmitting device must repeat the current data in the next cycle when it detects that PSSI_RDY is inactive. This flow control function allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

![Figure 340. Ready in receive mode waveform diagram (CKPOL=0)](image)

If the PSSI_RDY alternate input function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI samples the PSSI_RDY signal on the opposite PSSI_PDCK edge to the one at which D[15:0] are driven. If the PSSI_RDY signal is inactive, the PSSI keeps the same data (D[15:0]) and PSSI_DE signals that valid data are available during the next PSSI_PDCK clock cycle. Otherwise, if PSSI_RDY signal is sampled as active, the next data from the TX FIFO (if available) is output on the data outputs (D[15:0]). If no new data are available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI_DE signal as inactive (if enabled).

The receiving device uses the PSSI_RDY to control the data flow and avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

**Bidirectional PSSI_DE/PSSI_RDY signal**

A single pin can be used for both data enable (PSSI_DE) and ready (PSSI_RDY) functions if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the PSSI_CR register. In this case, the GPIO corresponding to selected alternate function (PSSI_DE when DERDYCFG=111 or PSSI_RDY when DERDYCFG=100) must be configured as open-drain. The other device must also be configured to drive the line as open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling the current data), then both devices know that the data is not being transferred in the current cycle.
32.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to Table 303 for the list of interrupts.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from low-power mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT_OVR</td>
<td>indicates overrun in receive mode or underrun in transmit mode</td>
<td>OVR_RIS</td>
<td>OVR_IE</td>
<td>OVR_ISC</td>
<td>NA</td>
</tr>
</tbody>
</table>

32.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR, results in a bus error. 32-bit read and write operations are permitted.

32.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00
Parallel synchronous slave interface (PSSI)

Reset value: 0x4000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>OUTEN: Data direction selection bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Receive mode: data is input synchronously with PSSI_PDCK</td>
</tr>
<tr>
<td>1</td>
<td>Transmit mode: data is output synchronously with PSSI_PDCK</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>DMAEN: DMA enable bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.</td>
</tr>
<tr>
<td>1</td>
<td>DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>CKSRC: Clock source</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>External clock (PSSI_PDCK in input)</td>
</tr>
<tr>
<td>1</td>
<td>Internal clock (PSSI_PDCK in output)</td>
</tr>
</tbody>
</table>

| Bits 28:21 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 20:18</th>
<th>DERDYCFG[2:0]: Data enable and ready configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>PSSI_DE and PSSI_RDY both disabled</td>
</tr>
<tr>
<td>001</td>
<td>Only PSSI_RDY enabled</td>
</tr>
<tr>
<td>010</td>
<td>Only PSSI_DE enabled</td>
</tr>
<tr>
<td>011</td>
<td>Both PSSI_RDY and PSSI_DE alternate functions enabled</td>
</tr>
<tr>
<td>100</td>
<td>Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see Bidirectional PSSI_DE/PSSI_RDY signal on page 1367)</td>
</tr>
<tr>
<td>101</td>
<td>Only PSSI_RDY function enabled, but mapped to PSSI_DE pin</td>
</tr>
<tr>
<td>110</td>
<td>Only PSSI_DE function enabled, but mapped to PSSI_RDY pin</td>
</tr>
<tr>
<td>111</td>
<td>Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see Bidirectional PSSI_DE/PSSI_RDY signal on page 1367)</td>
</tr>
</tbody>
</table>

When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

| Bits 17:15 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>ENABLE: PSSI enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PSSI disabled</td>
</tr>
<tr>
<td>1</td>
<td>PSSI enabled</td>
</tr>
</tbody>
</table>

The contents of the FiFO are flushed when ENABLE is cleared to 0.

**Note:** When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.

The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.
### Parallel synchronous slave interface (PSSI)

#### 32.5.2 PSSI status register (PSSI_SRM)

**Address offset:** 0x04  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

**Bits 13:12** Reserved, must be kept at reset value.

**Bits 11:10** **EDM[1:0]:** Extended data mode
- 00: Interface captures 8-bit data on every parallel data clock
- 01: Reserved, must not be selected
- 10: Reserved, must not be selected
- 11: The interface captures 16-bit data on every parallel data clock

**Bit 9** Reserved, must be kept at reset value.

**Bit 8** **RDYPOL:** Ready (PSSI_RDY) polarity
- This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.
- 0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)
- 1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

**Bit 7** Reserved, must be kept at reset value.

**Bit 6** **DEPOL:** Data enable (PSSI_DE) polarity
- This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.
- 0: PSSI_DE active low (0 indicates that data is valid)
- 1: PSSI_DE active high (1 indicates that data is valid)

**Bit 5** **CKPOL:** Parallel data clock polarity
- This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.
- 0: Falling edge active for inputs or rising edge active for outputs
- 1: Rising edge active for inputs or falling edge active for outputs

**Bits 4:0** Reserved, must be kept at reset value.
32.5.3 **PSSI raw interrupt status register (PSSI_RIS)**

Address offset: 0x08

Reset value: 0x0000 0000

PSSI_RIS gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 1 **OVR_RIS**: Data buffer overrun/underrun raw interrupt status

0: No overrun/underrun occurred
1: An overrun/underrun occurred: overrun in receive mode, underrun in transmit mode.

This bit is cleared by writing a 1 to the OVR_ISC bit in PSSI_ICR.

Bit 0 Reserved, must be kept at reset value.
### 32.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

<table>
<thead>
<tr>
<th>Bit 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td><strong>OVR_IE</strong>: Data buffer overrun/underrun interrupt enable</td>
</tr>
<tr>
<td>0</td>
<td>No interrupt generation</td>
</tr>
<tr>
<td>1</td>
<td>An interrupt is generated if either an overrun or an underrun error occurred.</td>
</tr>
</tbody>
</table>

| Bit 0    | Reserved, must be kept at reset value. |

### 32.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td><strong>OVR_MIS</strong>: Data buffer overrun/underrun masked interrupt status</td>
</tr>
<tr>
<td>0</td>
<td>No interrupt is generated when an overrun/underrun error occurs</td>
</tr>
<tr>
<td>1</td>
<td>An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.</td>
</tr>
</tbody>
</table>

| Bit 0    | Reserved, must be kept at reset value. |
32.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14
Reset value: 0x0000 0000

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 1 **OVR_ISC**: Data buffer overrun/underrun interrupt status clear
   Writing this bit to 1 clears the OVR_RIS bit in PSSI_RIS.

Bit 0 Reserved, must be kept at reset value.

32.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28
Reset value: 0x0000 0000

In receive mode (OUTEN = 0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN = 1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 results in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.
32.5.8 PSSI register map

Table 304. PSSI register map and reset values

| Offset | Register name reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | PSSI_CR                  | G0 | G1 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|        |                          |    |    | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | PSSI_SR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x08   | PSSI_RIS                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | PSSI_IER                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | PSSI_MIS                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x14   | PSSI_ICR                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x18-  | Reserved                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   |                          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x28   | PSSI_DR                  | BYTE3[7:0] | BYTE2[7:0] | BYTE1[7:0] | BYTE0[7:0] |
|        |                          | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 for the register boundary addresses.
33 LCD-TFT display controller (LTDC)

33.1 Introduction

The LCD-TFT (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (red, green, blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD and TFT panels.

33.2 LTDC main features

- 24-bit RGB parallel pixel output; 8 bits-per-pixel (RGB888)
- 2 display layers with dedicated FIFO (64x64-bit)
- Color look-up table (CLUT) up to 256 color (256x24-bit) per layer
- Programmable timings for different display panels
- Programmable background color
- Programmable polarity for HSYNC, VSYNC and data enable
- Up to 8 input color formats selectable per layer:
  - ARGB8888
  - RGB888
  - RGB565
  - ARGB1555
  - ARGB4444
  - L8 (8-bit luminance or CLUT)
  - AL44 (4-bit alpha + 4-bit luminance)
  - AL88 (8-bit alpha + 8-bit luminance)
- Pseudo-random dithering output for low bits per channel
  - Dither width 2 bits for red, green, blue
- Flexible blending between two layers using alpha value (per pixel or constant)
- Color keying (transparency color)
- Programmable window position and size
- Supports thin film transistor (TFT) color displays
- AXI master interface with burst of 16 double-words
- Up to 4 programmable interrupt events
33.3 LTDC functional description

33.3.1 LTDC block diagram

Figure 343. LTDC block diagram

33.3.2 LTDC pins and internal signals

The table below summarizes the LTDC signal interface.

<table>
<thead>
<tr>
<th>LCD-TFT signals</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LCD_CLK</td>
<td>Output</td>
<td>Clock output</td>
</tr>
<tr>
<td>LCD_HSYNC</td>
<td>Output</td>
<td>Horizontal synchronization</td>
</tr>
<tr>
<td>LCD_VSYNC</td>
<td>Output</td>
<td>Vertical synchronization</td>
</tr>
<tr>
<td>LCD_DE</td>
<td>Output</td>
<td>Not data enable</td>
</tr>
<tr>
<td>LCD_R[7:0]</td>
<td>Output</td>
<td>8-bit Red data</td>
</tr>
<tr>
<td>LCD_G[7:0]</td>
<td>Output</td>
<td>8-bit Green data</td>
</tr>
<tr>
<td>LCD_B[7:0]</td>
<td>Output</td>
<td>8-bit Blue data</td>
</tr>
</tbody>
</table>

The LTDC pins must be configured by the user application. The unused pins can be used for other purposes.

For LTDC outputs up to 24 bits (RGB888), if less than 8 bpp are used to output for example RGB565 or RGB666 to interface on 16- or 18-bit displays, the RGB display data lines must be connected to the MSB of the LTDC RGB data lines.

As an example, in the case of an LTDC interfacing with a RGB565 16-bit display, the LTDC display R[4:0], G[5:0] and B[4:0] data lines pins must be connected to the LCD_R[7:3], LCD_G[7:2] and LCD_B[7:3] pins.
The internal signals of the LTDC are given in the table below.

### Table 306. LTDC internal signals

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ltdc_aclk</td>
<td>Input</td>
<td>LTDC AXI clock</td>
</tr>
<tr>
<td>ltdc_pclk</td>
<td>Input</td>
<td>LTDC APB clock for register access</td>
</tr>
<tr>
<td>ltdc_ker_ck</td>
<td>Input</td>
<td>LTDC kernel clock used for LCD_CLK (pixel clock) generation</td>
</tr>
<tr>
<td>ltdc_it</td>
<td>Output</td>
<td>LTDC global interrupt request</td>
</tr>
<tr>
<td>ltdc_err_it</td>
<td>Output</td>
<td>LTDC global error interrupt request</td>
</tr>
<tr>
<td>ltdc_li</td>
<td>Output</td>
<td>LTDC line interrupt flag</td>
</tr>
</tbody>
</table>

The table below shows how the LTDC flags are connected.

### Table 307. LTDC trigger interconnection

<table>
<thead>
<tr>
<th>Trigger name</th>
<th>Direction</th>
<th>Trigger source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>ltdc_li</td>
<td>Output</td>
<td>hpdma_trigsel[12]</td>
</tr>
</tbody>
</table>

#### 33.3.3 LTDC reset and clocks

The LTDC controller peripheral uses the following clock domains:
- **AXI clock domain (ltdc_aclk)**
  This domain contains the LTDC AXI master interface for data transfer from the memories to the layer FIFO and the frame-buffer configuration register.
- **APB clock domain (ltdc_pclk)**
  This domain contains the global configuration registers and the interrupt register.
- **Pixel clock domain (LCD_CLK)**
  This domain contains the pixel data generation, the layer configuration register as well as the LTDC interface signal generator. The LCD_CLK output must be configured following the panel requirements. The LCD_CLK is generated from a specific PLL output (refer to the reset and clock control section).

The table below summarizes the clock domain for each register.

### Table 308. Clock domain for each register

<table>
<thead>
<tr>
<th>LTDC register</th>
<th>Clock domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTDC_LxCR</td>
<td>ltdc_aclk</td>
</tr>
<tr>
<td>LTDC_LxCFBAR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCFBLR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCFBLNR</td>
<td></td>
</tr>
</tbody>
</table>
Care must be taken while accessing the LTDC registers, the APB bus is stalled during the access for a given time period (see the table below).

<table>
<thead>
<tr>
<th>LTDC register</th>
<th>Clock domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTDC_SRCR</td>
<td>ltdc_pclk</td>
</tr>
<tr>
<td>LTDC_IER</td>
<td></td>
</tr>
<tr>
<td>LTDC_ISR</td>
<td></td>
</tr>
<tr>
<td>LTDC_ICR</td>
<td></td>
</tr>
<tr>
<td>LTDC_SSCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_BPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_AWCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_TWCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_GCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_BCCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LIPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_CPSR</td>
<td></td>
</tr>
<tr>
<td>LTDC_CDSR</td>
<td>Pixel clock (LCD_CLK)</td>
</tr>
<tr>
<td>LTDC_LxWHPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxWVPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCKCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxFPCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCACR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxDCCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxBFCR</td>
<td></td>
</tr>
<tr>
<td>LTDC_LxCLUTWR</td>
<td></td>
</tr>
</tbody>
</table>

The LTDC controller can be reset by setting the corresponding bit in the RCC. It resets the three clock domains.
33.4 **LTDC programmable parameters**

The LTDC controller provides flexible configurable parameters. It can be enabled or disabled through the LTDC_GCR register.

33.4.1 **LTDC global configuration parameters**

**Synchronous timings**

The figure below presents the configurable timing parameters generated by the synchronous timings generator block presented in the block diagram *Figure 343*. It generates the horizontal and vertical synchronization timings panel signals, the pixel clock and the data enable signals.

*Figure 344. LTDC synchronous timings*

The LTDC programmable synchronous timings are the following:

- **HSYNC and VSYNC width**: horizontal and vertical synchronization width, configured by programming a value of HSYNC width - 1 and VSYNC width - 1 in the LTDC_SSCR register
- **HBP and VBP**: horizontal and vertical synchronization back porch width, configured by programming the accumulated value HSYNC width + HBP - 1 and the accumulated value VSYNC width + VBP - 1 in the LTDC_BPCR register.
- **Active width and active height**: the active width and active height are configured by programming the accumulated value HSYNC width + HBP + active width - 1 and the
accumulated value VSYNC width + VBP + active height - 1 in the LTDC_AWCR register.

- Total width: the total width is configured by programming the accumulated value HSYNC width + HBP + active width + HFP - 1 in the LTDC_TWCR register. The HFP is the horizontal front porch period.
- Total height: the total height is configured by programming the accumulated value VSYNC height + VBP + active height + VFP - 1 in the LTDC_TWCR register. The VFP is the vertical front porch period.

Note: When the LTDC is enabled, the timings generated start with X/Y = 0/0 position as the first horizontal synchronization pixel in the vertical synchronization area and following the back porch, active data display area and the front porch.
When the LTDC is disabled, the timing generator block is reset to X = total width - 1, Y = total height - 1 and held the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore only blanking data is output continuously.

Example of synchronous timings configuration

LTDC timings (must be extracted from panel datasheet):
- horizontal and vertical synchronization width: 0xA pixels and 0x2 lines
- horizontal and vertical back porch: 0x14 pixels and 0x2 lines
- active width and active height: 0x140 pixels, 0xF0 lines (320x240)
- horizontal front porch: 0xA pixels
- vertical front porch: 0x4 lines

The programmed values in the LTDC timings registers are:
- LTDC_SSCR register to be programmed to 0x00090001 (HSW[11:0] is 0x9 and VSH[10:0] is 0x1)
- LTDC_BPCR register to be programmed to 0x001D0003 (AHBP[11:0] is 0x1D (0xA+ 0x13) and AVBP[10:0]A is 0x3 (0x2 + 0x1))
- LTDC_AWCR register to be programmed to 0x015D00F3 (AAW[11:0] is 0x15D (0xA +0x14 +0x13F) and AAH[10:0] is 0xF3 (0x2 + 0x2 + 0xEF))
- LTDC_TWCR register to be programmed to 0x000000167 (TOTALW[11:0] is 0x167 (0xA +0x14 +0x140 + 0x9))
- LTDC_THCR register to be programmed to 0x000000F7 (TOTALH[10:0]is 0xF7 (0x2 +0x2 + 0xF0 + 3))

Programmable polarity

The horizontal and vertical synchronization, data enable and pixel clock output signals polarity can be programmed to active high or active low through the LTDC_GCR register.

Background color

A constant background color (RGB888) can programmed through the LTDC_BCCR register. It is used for blending with the bottom layer.

Dithering

The dithering pseudo-random technique using an LFSR is used to add a small random value (threshold) to each pixel color channel (R, G or B) value, thus rounding up the MSB in
some cases when displaying a 24-bit data on 18-bit display. Thus the dithering technique is used to round data which is different from one frame to the other.

The dithering pseudo-random technique is the same as comparing LSBs against a threshold value and adding a 1 to the MSB part only, if the LSB part is ≥ the threshold. The LSBs are typically dropped once dithering was applied.

The width of the added pseudo-random value is two bits for each color channel: two bits for red, two bits for green and two bits for blue.

Once the LTDC is enabled, the LFSR starts running with the first active pixel and it is kept running even during blanking periods and when dithering is switched off. If the LTDC is disabled, the LFSR is reset.

The dithering can be switched on and off on the fly through the LTDC_GCR register.

**Reload shadow registers**

Some configuration registers are shadowed. The shadow registers values can be reloaded immediately to the active registers when writing to these registers or at the beginning of the vertical blanking period following the configuration in the LTDC_SRCR register. If the immediate reload configuration is selected, the reload must be activated only when all new registers have been written.

The shadow registers must not be modified again before the reload is done. Reading from the shadow registers returns the actual active value. The new written value can only be read after the reload has taken place.

A register reload interrupt can be generated if enabled in the LTDC_IER register.

The shadowed registers are all Layer1 and Layer2 registers except LTDC_LxCLUTWR.

**Interrupt generation event**

Refer to [Section 33.5: LTDC interrupts](#) for the interrupt configuration.

### 33.4.2 Layer programmable parameters

Up to two layers can be enabled, disabled and configured separately. The layer display order is fixed and it is bottom up. If two layers are enabled, the layer2 is the top displayed window.

**Windowing**

Every layer can be positioned and resized and it must be inside the active display area.

The window position and size are configured through the top-left and bottom-right X/Y positions and the internal timing generator that includes the synchronous, back porch size and the active data area. Refer to LTDC_LxWHPCR and LTDC_WVPCR registers.

The programmable layer position and size defines the first/last visible pixel of a line and the first/last visible line in the window. It allows to display either the full image frame or only a part of the image frame (see [Figure 345](#)):

- The first and the last visible pixel in the layer are set by configuring the WHSTPOS[11:0] and WHSPPOS[11:0] in the LTDC_LxWHPCR register.
- The first and the last visible lines in the layer are set by configuring the WVSTPOS[10:0] and WVSPPOS[10:0] in the LTDC_LxWVPCR register.
Pixel input format

The programmable pixel format is used for the data stored in the frame buffer of a layer. Up to eight input pixel formats can be configured for every layer through the LTDC_LxPFCR register.

The pixel data is read from the frame buffer and then transformed to the internal 8888 (ARGB) format as follows: components having a width of less than 8 bits get expanded to 8 bits by bit replication. The selected bit range is concatenated multiple times until it is longer than 8 bits. Of the resulting vector, the 8 MSB bits are chosen. Example: 5 bits of an RGB565 red channel become (bit positions) 43210432 (the three LSBs are filled with the three MSBs of the five bits).

The table below describes the pixel data mapping depending on the selected format.

Table 310. Pixel data mapping versus color format

<table>
<thead>
<tr>
<th>ARGB8888</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>@+3 A_x[7:0]</td>
<td>@+2 R_x[7:0]</td>
<td>@+1 G_x[7:0]</td>
<td>@ B_x[7:0]</td>
<td></td>
</tr>
<tr>
<td>@+7 A_x+[7:0]</td>
<td>@+6 R_x+[7:0]</td>
<td>@+5 G_x+[7:0]</td>
<td>@+4 B_x+[7:0]</td>
<td></td>
</tr>
<tr>
<td>RGB888</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@+3 B_x+[7:0]</td>
<td>@+2 R_x[7:0]</td>
<td>@+1 G_x[7:0]</td>
<td>@ B_x[7:0]</td>
<td></td>
</tr>
<tr>
<td>@+7 G_x+[7:0]</td>
<td>@+6 B_x+[7:0]</td>
<td>@+5 R_x+[7:0]</td>
<td>@+4 G_x+[7:0]</td>
<td></td>
</tr>
<tr>
<td>RGB565</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>@+3 R_x+[4:0] G_x+[5:3]</td>
<td>@+2 G_x+[2:0] B_x+[4:0]</td>
<td>@+1 R_x+[4:0] G_x+[5:3]</td>
<td>@ G_x+[2:0] B_x+[4:0]</td>
<td></td>
</tr>
<tr>
<td>ARGB1555</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The CLUT can be enabled at run-time for every layer through the LTDC_LxCR register and it is only useful in case of indexed color when using the L8, AL44 and AL88 input pixel format.

First, the CLUT must be loaded with the R, G and B values that replace the original R, G, B values of that pixel (indexed color). Each color (RGB value) has its own address that is the position within the CLUT.

The R, G and B values and their own respective address are programmed through the LTDC_LxCLUTWR register:
- In case of L8 and AL88 input pixel format, the CLUT must be loaded by 256 colors. The address of each color is configured in the CLUTADD bits in the LTDC_LxCLUTWR register.

### Table 310. Pixel data mapping versus color format (continued)

<table>
<thead>
<tr>
<th>Address</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>@+3 A_{x}[7:3] B_{x}[3:0] L_{x}[7:0]</td>
<td>L8</td>
</tr>
<tr>
<td>@+7 A_{x}[7:3] B_{x}[3:0] L_{x}[7:0]</td>
<td>AL44</td>
</tr>
<tr>
<td>@+3 A_{x}[7:3] B_{x}[3:0] L_{x}[7:0]</td>
<td>AL88</td>
</tr>
</tbody>
</table>

**Color look-up table (CLUT)**

The CLUT can be enabled at run-time for every layer through the LTDC_LxCR register and it is only useful in case of indexed color when using the L8, AL44 and AL88 input pixel format.

First, the CLUT must be loaded with the R, G and B values that replace the original R, G, B values of that pixel (indexed color). Each color (RGB value) has its own address that is the position within the CLUT.

The R, G and B values and their own respective address are programmed through the LTDC_LxCLUTWR register:
- In case of L8 and AL88 input pixel format, the CLUT must be loaded by 256 colors. The address of each color is configured in the CLUTADD bits in the LTDC_LxCLUTWR register.
In case of AL44 input pixel format, the CLUT must be loaded by only 16 colors. The address of each color must be filled by replicating the 4-bit L channel to 8-bit as follows:

- L0 (indexed color 0), at address 0x00
- L1, at address 0x11
- L2, at address 0x22
- ......
- L15, at address 0xFF

**Color frame buffer address**

Every layer has a start address for the color frame buffer configured through the LTDC_LxCFBAR register.

When a layer is enabled, the data is fetched from the color frame buffer.

**Color frame buffer length**

Every layer has a total line length setting for the color frame buffer in bytes and a number of lines in the frame buffer configurable in the LTDC_LxCFBLR and LTDC_LxCFBLNR register respectively.

The line length and the number of lines settings are used to stop the prefetching of data to the layer FIFO at the end of the frame buffer:

- If it is set to less bytes than required, a FIFO underrun interrupt is generated if it has been previously enabled.
- If it is set to more bytes than actually required, the useless data read from the FIFO is discarded. The useless data is not displayed.

**Color frame buffer pitch**

Every layer has a configurable pitch for the color frame buffer, that is the distance between the start of one line and the beginning of the next line in bytes. It is configured through the LTDC_LxCFBLR register.

**Layer blending**

The blending is always active and the two layers can be blended following the blending factors configured through the LTDC_LxBFCR register.

The blending order is fixed and it is bottom up. If two layers are enabled, first the Layer 1 is blended with the Background color, then the layer 2 is blended with the result of blended color of layer 1 and the background. Refer to the figure below.

**Figure 346. Blending two layers with background**
**Default color**

Every layer can have a default color in the format ARGB which is used outside the defined layer window or when a layer is disabled.

The default color is configured through the LTDC_LxDCCR register.

The blending is always performed between the two layers even when a layer is disabled. To avoid displaying the default color when a layer is disabled, keep the blending factors of this layer in the LTDC_LxBFCR register to their reset value.

**Color keying**

A color key (RGB) can be configured to be representative for a transparent pixel.

If the color keying is enabled, the current pixels (after format conversion and before CLUT respectively blending) are compared to the color key. If they match for the programmed RGB value, all channels (ARGB) of that pixel are set to 0.

The color key value can be configured and used at run-time to replace the pixel RGB value.

The color keying is enabled through the LTDC_LxCKCR register.

The color keying is configured through the LTDC_LxCKCR register. The programmed value depends on the pixel format as it is compared to current pixel after pixel format conversion to ARGB888.

**Example**: if the a mid-yellow color (50 % red + 50 % green) is used as the transparent color key:

- In RGB565, the mid-yellow color is 0x8400. Set the LTDC_LxCKCR to 0x848200.
- In ARGB8888, the mid-yellow color is 0x808000. Set LTDC_LxCKCR to 0x808000.
- In all CLUT-based color modes (L8, AL88, AL44), set one of the palette entry to the mid-yellow color 0x808000 and set the LTDC_LxCKCR to 0x808000.

**33.5 LTDC interrupts**

The LTDC provides four maskable interrupts logically ORed to two interrupt vectors.

The interrupt sources can be enabled or disabled separately through the LTDC_IER register. Setting the appropriate mask bit to 1 enables the corresponding interrupt.

The two interrupts are generated on the following events:

- Line interrupt: generated when a programmed line is reached. The line interrupt position is programmed in the LTDC_LiPCR register
- Register reload interrupt: generated when the shadow registers reload is performed during the vertical blanking period
- FIFO underrun interrupt: generated when a pixel is requested from an empty layer FIFO
- Transfer error interrupt: generated when an AXI bus error occurs during data transfer
These interrupt events are connected to the NVIC controller as described in the figure below.

**Figure 347. Interrupt events**

![Diagram of interrupt events](image)

**Table 311. LTDC interrupt requests**

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Line</td>
<td>LIF</td>
<td>LIE</td>
</tr>
<tr>
<td>Register reload</td>
<td>RRIF</td>
<td>RRIEN</td>
</tr>
<tr>
<td>FIFO underrun</td>
<td>FUDERRIF</td>
<td>FUDERRIE</td>
</tr>
<tr>
<td>Transfer error</td>
<td>TERRIF</td>
<td>TERRIE</td>
</tr>
</tbody>
</table>

### 33.6 LTDC programming procedure

The steps listed below are needed to program the LTDC:

1. Enable the LTDC clock in the RCC register.
2. Configure the required pixel clock following the panel datasheet.
3. Configure the synchronous timings: VSYNC, HSYNC, vertical and horizontal back porch, active data area and the front porch timings following the panel datasheet as described in the **Section 33.4.1**.
4. Configure the synchronous signals and clock polarity in the LTDC_GCR register.
5. If needed, configure the background color in the LTDC_BCCR register.
6. Configure the needed interrupts in the LTDC_IER and LTDC_LIPCR register.
7. Configure the layer1/2 parameters by:
   - programming the layer window horizontal and vertical position in the LTDC_LxWHPCR and LTDC_IWVPCR registers. The layer window must be in the active data area.
   - programming the pixel input format in the LTDC_LxPFPCR register
   - programming the color frame buffer start address in the LTDC_LxCFBAR register
   - programming the line length and pitch of the color frame buffer in the LTDC_LxCFBLR register
   - programming the number of lines of the color frame buffer in the LTDC_LxCFBLNR register
   - if needed, loading the CLUT with the RGB values and its address in the LTDC_LxCLUTWR register
   - If needed, configuring the default color and the blending factors respectively in the LTDC_LxDCCR and LTDC_LxBFCR registers
8. Enable layer1/2 and if needed the CLUT in the LTDC_LxCR register.
9. If needed, enable dithering and color keying respectively in the LTDC_GCR and LTDC_LxCKCR registers. They can be also enabled on the fly.
10. Reload the shadow registers to active register through the LTDC_SRCR register.
11. Enable the LTDC controller in the LTDC_GCR register.
12. All layer parameters can be modified on the fly except the CLUT. The new configuration must be either reloaded immediately or during vertical blanking period by configuring the LTDC_SRCR register.

Note: All layer’s registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. Thus, a new write to the same register overrides the previous configuration if not yet reloaded.

### 33.7 LTDC registers

#### 33.7.1 LTDC synchronization size configuration register (LTDC_SSCR)

Address offset: 0x008
Reset value: 0x0000 0000

This register defines the number of horizontal synchronization pixels minus 1 and the number of vertical synchronization lines minus 1. Refer to Figure 344 and Section 33.4 for an example of configuration.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27:16</td>
<td>HSW[11:0]: Horizontal synchronization width (in units of pixel clock period)</td>
</tr>
<tr>
<td>15:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10:0</td>
<td>VSH[10:0]: Vertical synchronization height (in units of horizontal scan line)</td>
</tr>
</tbody>
</table>

#### 33.7.2 LTDC back porch configuration register (LTDC_BPCR)

Address offset: 0x00C
Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (HSYNC width + HBP - 1) and the accumulated number of vertical
synchronization and back porch lines minus 1 (VSYNC height + VBP - 1).
Refer to Figure 344 and Section 33.4 for an example of configuration.

33.7.3 LTDC active width configuration register (LTDC_AWCR)

Address offset: 0x010
Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (HSYNC width + HBP + active width - 1) and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (VSYNC height + VBP + active height - 1). Refer to Figure 344 and Section 33.4 for an example of configuration.

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 AHW[11:0]: Accumulated horizontal back porch (in units of pixel clock period)
These bits defines the accumulated horizontal back porch width that includes the horizontal synchronization and horizontal back porch pixels minus 1.
The horizontal back porch is the period between horizontal synchronization going inactive and the start of the active display part of the next scan line.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 AVBP[10:0]: Accumulated Vertical back porch (in units of horizontal scan line)
The vertical back porch is the number of horizontal scan lines at a start of frame to the start of the first active scan line of the next frame.

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 AAW[11:0]: Accumulated active width (in units of pixel clock period)
These bits define the accumulated active width which includes the horizontal synchronization, horizontal back porch and active pixels minus 1.
The active width is the number of pixels in active display area of the panel scan line.
Refer to device datasheet for maximum active width supported following maximum pixel clock.
33.7.4 LTDC total width configuration register (LTDC_TWCR)

Address offset: 0x014
Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (HSYNC width + HBP + active width + HFP - 1) and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (VSYNC height + VBP + active height + VFP - 1). Refer to Figure 344 and Section 33.4 for an example of configuration.

| Bits 31:28 | Reserved, must be kept at reset value. |
| Bits 27:16 | TOTALW[11:0]: Total width (in units of pixel clock period) |
| Bits 15:11 | Reserved, must be kept at reset value. |
| Bits 10:0  | TOTALH[10:0]: Total height (in units of horizontal scan line) |

33.7.5 LTDC global control register (LTDC_GCR)

Address offset: 0x018
Reset value: 0x0000 2220

This register defines the global configuration of the LCD-TFT controller.

| Bits 31:28 | Reserved, must be kept at reset value. |
| Bits 27:16 | TOTALW[11:0]: Total width (in units of pixel clock period) |
| Bits 15:11 | Reserved, must be kept at reset value. |
| Bits 10:0  | TOTALH[10:0]: Total height (in units of horizontal scan line) |

ST
Bit 31 **HSPOL**: Horizontal synchronization polarity
   This bit is set and cleared by software.
   0: Horizontal synchronization polarity is active low.
   1: Horizontal synchronization polarity is active high.

Bit 30 **VSPOL**: Vertical synchronization polarity
   This bit is set and cleared by software.
   0: Vertical synchronization is active low.
   1: Vertical synchronization is active high.

Bit 29 **DEPOL**: Not data enable polarity
   This bit is set and cleared by software.
   0: Not data enable polarity is active low.
   1: Not data enable polarity is active high.

Bit 28 **PCPOL**: Pixel clock polarity
   This bit is set and cleared by software.
   0: Pixel clock polarity is active low.
   1: Pixel clock is active high.

Bits 27:17 Reserved, must be kept at reset value.

Bit 16 **DEN**: Dither enable
   This bit is set and cleared by software.
   0: Dither disabled
   1: Dither enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DRW[2:0]**: Dither red width
   These bits return the dither red bits.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DGW[2:0]**: Dither green width
   These bits return the dither green bits.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DBW[2:0]**: Dither blue width
   These bits return the dither blue bits.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **LTDCEN**: LCD-TFT controller enable
   This bit is set and cleared by software.
   0: LTDC disabled
   1: LTDC enabled
33.7.6 LTDC shadow reload configuration register (LTDC_SRCR)

Address offset: 0x024
Reset value: 0x0000 0000

This register allows to reload either immediately or during the vertical blanking period, the shadow registers values to the active registers. The shadow registers are all Layer1 and Layer2 registers except the LTDC_L1CLUTWR and the LTDC_L2CLUTWR.

The shadow registers read back the active values. Until the reload has been done, the 'old' value is read.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
</tbody>
</table>

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

Bit 1 **VBR**: Vertical blanking reload
This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).
0: No effect
1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: Immediate reload
This bit is set by software and cleared only by hardware after reload.
0: No effect
1: The shadow registers are reloaded immediately.

33.7.7 LTDC background color configuration register (LTDC_BCCR)

Address offset: 0x02C
Reset value: 0x0000 0000

This register defines the background color (RGB888).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
<td>Rw</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BCRED[7:0]**: Background color red value
These bits configure the background red value.
33.7.8 **LTDC interrupt enable register (LTDC_IER)**

Address offset: 0x034  
Reset value: 0x0000 0000  
This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 3 **RRIE**: Register reload interrupt enable  
This bit is set and cleared by software.  
0: Register reload interrupt disable  
1: Register reload interrupt enable

Bit 2 **TERRIE**: Transfer error interrupt enable  
This bit is set and cleared by software.  
0: Transfer error interrupt disable  
1: Transfer error interrupt enable

Bit 1 **FUIE**: FIFO underrun interrupt enable  
This bit is set and cleared by software.  
0: FIFO underrun interrupt disable  
1: FIFO underrun interrupt enable

Bit 0 **LIE**: Line interrupt enable  
This bit is set and cleared by software.  
0: Line interrupt disable  
1: Line interrupt enable
### 33.7.9 LTDC interrupt status register (LTDC_ISR)

Address offset: 0x038  
Reset value: 0x0000 0000  

This register returns the interrupt status flag.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RRIF</td>
<td>TERRIF</td>
<td>FUIF</td>
<td>LIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 3** **RRIF**: Register reload interrupt flag  
  0: No register reload interrupt generated  
  1: Register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

- **Bit 2** **TERRIF**: Transfer error interrupt flag  
  0: No transfer error interrupt generated  
  1: Transfer error interrupt generated when a bus error occurs

- **Bit 1** **FUIF**: FIFO underrun interrupt flag  
  0: No FIFO underrun interrupt generated  
  1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and pixel data is read from the FIFO

- **Bit 0** **LIF**: Line interrupt flag  
  0: No line interrupt generated  
  1: Line interrupt generated when a programmed line is reached

### 33.7.10 LTDC interrupt clear register (LTDC_ICR)

Address offset: 0x03C  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>CRRIF</td>
<td>CTERRIF</td>
<td>CFUIF</td>
<td>CLIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 3** **CRRIF**: Clear register reload interrupt flag  
  0: No effect  
  1: Clear the RRIF flag in the LTDC_ISR register
Bit 2 **CTERRIF**: Clear the transfer error interrupt flag
0: No effect
1: Clear the TERRIF flag in the LTDC_ISR register.

Bit 1 **CFUIF**: Clear the FIFO underrun interrupt flag
0: No effect
1: Clear the FUDERRIF flag in the LTDC_ISR register.

Bit 0 **CLIF**: Clear the line interrupt flag
0: No effect
1: Clear the LIF flag in the LTDC_ISR register.

### 33.7.11 LTDC line interrupt position configuration register (LTDC_LIPCR)

Address offset: 0x040

Reset value: 0x0000 0000

This register defines the position of the line interrupt. The line value to be programmed depends on the timings parameters. Refer to *Figure 344*.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **LIPOS[10:0]**: Line interrupt position
These bits configure the line interrupt position.

### 33.7.12 LTDC current position status register (LTDC_CPSR)

Address offset: 0x044

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 **CXPOS[15:0]**: Current X position
These bits return the current X position.

Bits 15:0 **CYPOS[15:0]**: Current Y position
These bits return the current Y position.
### 33.7.13 LTDC current display status register (LTDC_CDSR)

Address offset: 0x048  
Reset value: 0x0000 000F  

This register returns the status of the current display phase which is controlled by the HSYNC, VSYNC, and horizontal/vertical DE signals. 

Example: if the current display phase is the vertical synchronization, the VSYNCS bit is set (active high). If the current display phase is the horizontal synchronization, the HSYNCS bit is active high.

The returned status does not depend on the configured polarity in the LTDC_GCR register, instead it returns the current active display phase.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>15</td>
<td>HSYNCs</td>
<td>0: Active low</td>
</tr>
<tr>
<td>14</td>
<td>VSYNCs</td>
<td>0: Active low</td>
</tr>
<tr>
<td>13</td>
<td>HDES</td>
<td>0: Active low</td>
</tr>
<tr>
<td>12</td>
<td>VDES</td>
<td>0: Active low</td>
</tr>
<tr>
<td>11</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>10</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>9</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>8</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>7</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>6</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>4</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>Reserved</td>
<td>R</td>
</tr>
<tr>
<td>0</td>
<td>Reserved</td>
<td>R</td>
</tr>
</tbody>
</table>

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

- **Bit 3 HSYNCs**: Horizontal synchronization display status  
  0: Active low  
  1: Active high  

- **Bit 2 VSYNCs**: Vertical synchronization display status  
  0: Active low  
  1: Active high  

- **Bit 1 HDES**: Horizontal data enable display status  
  0: Active low  
  1: Active high  

- **Bit 0 VDES**: Vertical data enable display status  
  0: Active low  
  1: Active high

### 33.7.14 LTDC layer x control register (LTDC_LxCR)

Address offset: 0x084 + 0x80 * (x - 1), (x = 1 to 2)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>RW</td>
</tr>
<tr>
<td>15</td>
<td>CLUTEn</td>
<td>R</td>
</tr>
<tr>
<td>14</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>13</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>12</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>11</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>10</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>9</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>8</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>7</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>6</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>5</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>4</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>3</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>2</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>1</td>
<td>Res</td>
<td>R</td>
</tr>
<tr>
<td>0</td>
<td>Res</td>
<td>R</td>
</tr>
</tbody>
</table>

ST
33.7.15 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR)

Address offset: 0x088 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the horizontal position (first and last pixel) of the layer 1 or 2 window.

The first visible pixel of a line is the programmed value of AHBP[11:0] bits + 1 in the LTDC_BPCR register.

The last visible pixel of a line is the programmed value of AAW[11:0] bits in the LTDC_AWCR register.

Example: The LTDC_BPCR register is configured to 0x000E0005 (AHBP[11:0] is 0xE) and the LTDC_AWCR register is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:

- layer window first pixel, WHSTPOS[11:0], must be programmed to 0x14 (0xE+1+0x5).
- layer window last pixel, WHSPPOS[11:0], must be programmed to 0x28A.

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

Bit 4 CLUTEN: Color look-up table enable
This bit is set and cleared by software.
0: Color look-up table disable
1: Color look-up table enable
The CLUT is only meaningful for L8, AL44 and AL88 pixel format. Refer to Color look-up table (CLUT)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 COLKEN: Color keying enable
This bit is set and cleared by software.
0: Color keying disable
1: Color keying enable

Bit 0 LEN: Layer enable
This bit is set and cleared by software.
0: Layer disabled
1: Layer enabled

Bits 27:16 WHSPPOS[11:0]: Window horizontal stop position
These bits configure the last visible pixel of a line of the layer window.
33.7.16  **LTDC layer x window vertical position configuration register (LTDC_LxWVPCR)**

Address offset: 0x08C + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the vertical position (first and last line) of the layer1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[10:0] bits + 1 in the register LTDC_BPCR register.

The last visible line of a frame is the programmed value of AAH[10:0] bits in the LTDC_AWCR register.

**Example:**

The LTDC_BPCR register is configured to 0x000E0005 (AVBP[10:0] is 0x5) and the LTDC_AWCR register is configured to 0x028E01E5 (AAH[10:0] is 0x1E5).

To configure the vertical position of a window size of 630x460, with vertical start offset of eight lines in the active data area:

- layer window first line, WVSTPOS[10:0], must be programmed to 0xE (0x5 + 1 + 0x8).
- layer window last line, WVSPPOS[10:0] must be programmed to 0x1DA.

| Bits 31:27 | Reserved, must be kept at reset value. |
| Bits 26:16 | **WVSPPOS[10:0]**: Window vertical stop position |
| These bits configure the last visible line of the layer window. |
| Bits 15:11 | Reserved, must be kept at reset value. |
| Bits 10:0  | **WVSTPOS[10:0]**: Window vertical start position |
| These bits configure the first visible line of the layer window. |
| WVSTPOS[10:0] must be ≤ AAH[10:0] bits (programmed in LTDC_AWCR register). |
33.7.17  **LTDC layer x color keying configuration register (LTDC_LxCKCR)**

Address offset: \(0x090 + 0x80 \times (x - 1), (x = 1 \text{ to } 2)\)

Reset value: 0x0000 0000

This register defines the color key value (RGB), that is used by the color keying.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CKRED[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CKGREEN[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CKBLUE[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **CKRED[7:0]**: Color key red value

Bits 15:8  **CKGREEN[7:0]**: Color key green value

Bits 7:0  **CKBLUE[7:0]**: Color key blue value

33.7.18  **LTDC layer x pixel format configuration register (LTDC_LxPFCR)**

Address offset: \(0x094 + 0x80 \times (x - 1), (x = 1 \text{ to } 2)\)

Reset value: 0x0000 0000

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PF[2:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 2:0  **PF[2:0]**: Pixel format

These bits configure the pixel format

000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
101: L8 (8-bit luminance)
110: AL44 (4-bit alpha, 4-bit luminance)
111: AL88 (8-bit alpha, 8-bit luminance)
33.7.19 LTDC layer x constant alpha configuration register (LTDC_LxCACR)

Address offset: 0x098 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 00FF

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending. Refer to LTDC_LxBFCR register.

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

Bits 7:0 CONSTA[7:0]: Constant alpha

These bits configure the constant alpha used for blending. The constant alpha is divided by 255 by hardware.

Example: if the programmed constant alpha is 0xFF, the constant alpha value is 255 / 255 = 1.

33.7.20 LTDC layer x default color configuration register (LTDC_LxDCCR)

Address offset: 0x09C + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

Bits 31:24 DCALPHA[7:0]: Default color alpha

These bits configure the default alpha value.

Bits 23:16 DCRED[7:0]: Default color red

These bits configure the default red value.

Bits 15:8 DCGREEN[7:0]: Default color green

These bits configure the default green value.

Bits 7:0 DCBLUE[7:0]: Default color blue

These bits configure the default blue value.
33.7.21 LTDC layer x blending factors configuration register (LTDC_LxBFCR)

Address offset: 0x0A0 + 0x80 * (x - 1), (x = 1 to 2)
Reset value: 0x0000 0607

This register defines the blending factors F1 and F2.

The general blending formula is: BC = BF1 x C + BF2 x Cs
- \( BC \) = blended color
- \( BF1 \) = blend factor 1
- \( C \) = current layer color
- \( BF2 \) = blend factor 2
- \( Cs \) = subjacent layers blended color

The constant alpha value, is the programmed value in LTDC_LxCACR divided by 255 by hardware.

Example: Only layer1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in LTDC_LxCACR is 240 (0xF0). Thus, the constant alpha value is 240 / 255 = 0.94. C: current layer color is 128. Cs: background color is 48. Layer1 is blended with the background color.

\[ BC = \text{constant alpha} \times C + (1 - \text{constant alpha}) \times Cs = 0.94 \times 128 + (1 - 0.94) \times 48 = 123. \]

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BF1[2:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BF2[2:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 BF1[2:0]: Blending factor 1
These bits select the blending factor F1.
100: constant alpha
110: pixel alpha x constant alpha
Others: Reserved

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

Bits 2:0 BF2[2:0]: blending factor 2
These bits select the blending factor F2
101: 1 - constant alpha
111: 1 - (pixel alpha x constant alpha)
Others: Reserved
33.7.22  **LTDC layer x color frame buffer address register**  
(\texttt{LTDC\_LxCFBAR})

Address offset: 0x0AC + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color frame buffer start address which has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  \texttt{CFBADD[31:0]}: Color frame buffer start address  
These bits define the color frame buffer start address.

33.7.23  **LTDC layer x color frame buffer length register**  
(\texttt{LTDC\_LxCFBLR})

Address offset: 0x0B0 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color frame buffer line length and pitch.

**Example:**
- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 3 = 960), where pitch = line length requires a value of 0x03C003C7 to be written into this register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:29  Reserved, must be kept at reset value.

Bits 28:16  \texttt{CFBP[12:0]}: Color frame buffer pitch in bytes  
These bits define the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.

Bits 15:13  Reserved, must be kept at reset value.
33.7.24 **LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR)**

Address offset: 0x0B4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the number of lines in the color frame buffer.

The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt is generated if enabled.

The start address and pitch settings on the other hand define the correct start of every line in memory.

```plaintext
<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **CFBLNBR[10:0]: Frame buffer line number**

These bits define the number of lines in the frame buffer that corresponds to the active high width.

33.7.25 **LTDC layer x CLUT write register (LTDC_LxCLUTWR)**

Address offset: 0x0C4 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the CLUT address and the RGB value.

The CLUT write register must be configured only during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in the LTDC_LxCR register.

The CLUT is only meaningful for L8, AL44 and AL88 pixel format.

```plaintext
<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**CLUTADD[7:0]**

```plaintext
|   |   |   |   |   |   |   |   |   |
| w | w | w | w | w | w | w | w |
| w | w | w | w | w | w | w | w |
| w | w | w | w | w | w | w | w |
| 15| 14| 13| 12| 11| 10|  9|  8|
```

**RED[7:0]**

```plaintext
|   |   |   |   |   |   |   |   |   |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
```

**GREEN[7:0]**

```
|   |   |   |   |   |   |   |   |   |
| w | w | w | w | w | w | w | w |
| w | w | w | w | w | w | w | w |
| w | w | w | w | w | w | w | w |
```

**BLUE[7:0]**

```
|   |   |   |   |   |   |   |   |   |
| w | w | w | w | w | w | w | w |
| w | w | w | w | w | w | w | w |
| w | w | w | w | w | w | w | w |
```
Bits 31:24 **CLUTADD[7:0]**: CLUT address
These bits configure the CLUT address (color position within the CLUT) of each RGB value.

Bits 23:16 **RED[7:0]**: Red value
These bits configure the red value.

Bits 15:8 **GREEN[7:0]**: Green value
These bits configure the green value.

Bits 7:0 **BLUE[7:0]**: Blue value
These bits configure the blue value.

### 33.7.26 LTDC register map

#### Table 312. LTDC register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x008  | LTDC_SSCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x00C  | LTDC_BPCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x010  | LTDC_AWCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x014  | LTDC_TWCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x018  | LTDC_GCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x024  | LTDC_SRCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x02C  | LTDC_BCCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x030  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x034  | LTDC_IER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x038  | LTDC_ISR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Table 312. LTDC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>CCR</th>
<th>CERR</th>
<th>CIF</th>
<th>CUIF</th>
<th>CUIF</th>
<th>CUIF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03C</td>
<td>LTDC_ICR</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x040</td>
<td>LTDC_LIPCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>LTDC_CPSR</td>
<td>CXPOS[15:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x048</td>
<td>LTDC_CDSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x084</td>
<td>LTDC_L1CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x088</td>
<td>LTDC_L1WHPCR</td>
<td>WHSPO[11:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x08C</td>
<td>LTDC_L1WPCR</td>
<td>WVSTOP[10:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x090</td>
<td>LTDC_L1CKCR</td>
<td>CKRED[7:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x094</td>
<td>LTDC_L1PFCR</td>
<td>PF[2:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x098</td>
<td>LTDC_L1CACR</td>
<td>CONSTA[7:0]</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0x09C</td>
<td>LTDC_L1DCCR</td>
<td>DCALPHA[7.0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0A0</td>
<td>LTDC_L1BFCR</td>
<td>BF1[2:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x0A4-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0AC</td>
<td>LTDC_L1CFBAR</td>
<td>CFBADD[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B0</td>
<td>LTDC_L1CFBLR</td>
<td>CFB[12:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B4</td>
<td>LTDC_L1CFBLNR</td>
<td>CFBLL[12:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 312. LTDC register map and reset values (continued)
Refer to Section 2.3 for the register boundary addresses.
34 JPEG codec (JPEG)

34.1 Introduction
The hardware 8-bit JPEG codec encodes uncompressed image data stream or decodes JPEG-compressed image data stream. It also fully manages JPEG headers.

34.2 JPEG codec main features
- High-speed fully-synchronous operation
- Configurable as encoder or decoder
- Single-clock-per-pixel encode/decode
- RGB, YCbCr, YCMK and BW (grayscale) image color space support
- 8-bit depth per image component at encode/decode
- JPEG header generator/parser with enable/disable
- Four programmable quantization tables
- Single-clock Huffman coding and decoding
- Fully-programmable Huffman tables (two AC and two DC)
- Fully-programmable minimum coded unit (MCU)
- Concurrent input and output data stream interfaces
34.3 JPEG codec block functional description

34.3.1 General description

The block diagram of the JPEG codec is shown in Figure 348.

Figure 348. JPEG codec block diagram

34.3.2 JPEG internal signals

Table 313 lists the JPEG internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>jpeg_hclk</td>
<td>Digital input</td>
<td>JPEG kernel and register interface clock</td>
</tr>
<tr>
<td>jpeg_it</td>
<td>Digital output</td>
<td>JPEG global interrupt</td>
</tr>
<tr>
<td>jpeg_iftrg</td>
<td>Digital output</td>
<td>JPEG input FIFO threshold for DMA trigger</td>
</tr>
<tr>
<td>jpeg_ifntf</td>
<td>Digital output</td>
<td>JPEG input FIFO not full for DMA trigger</td>
</tr>
<tr>
<td>jpeg_oftrg</td>
<td>Digital output</td>
<td>JPEG output FIFO threshold for DMA trigger</td>
</tr>
</tbody>
</table>
34.3.3 JPEG decoding procedure

The JPEG codec can decode a JPEG stream as defined in the ISO/IEC 10918-1 specification.

It can optionally parse the JPEG header and update accordingly the JPEG codec registers, the quantization tables and the Huffman tables.

The JPEG codec is configured in decode mode setting the DE bit (decode enable) of the JPEG_CONFR1 register.

The JPEG decode starts by setting the START bit of the JPEG_CONFR0 register.

The JPEG codec requests data for its input FIFO through generating one of:

- DMA request
- DMA trigger
- interrupts

**DMA generation for input FIFO**

DMA request is generated when the 32-byte input FIFO becomes at least half-empty, that is, when there is free room for writing 16 bytes of data.

The DMA request generation is independent of the START bit of the JPEG_CONFR0 register. If the input FIFO can accept 16 bytes and the DMA for the input FIFO is enabled (setting the IDMAEN bit of the JPEG_CR register), a DMA request is generated regardless of the state of the JPEG codec kernel.

A burst transfer is launched by the DMA to write 16 bytes of data.

Writes are ignored if the input FIFO is full.

At the end of the decoding process, extra bytes may remain in the input FIFO and/or a DMA request may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the input FIFO must be disabled to prevent unwanted DMA request upon flushing the FIFO.
The consequence of not flushing the FIFO at the end of the decoding process is that any remaining data is taken into the next JPEG decoding.

DMA requests are no more generated once the EOCF flag of the JPEG_SR register is set.

**Interrupt or DMA trigger generation for input FIFO**

Input FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:
- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or DMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the decoding process, extra bytes may remain in the input FIFO and/or an interrupt request / DMA trigger may be pending. The FIFO can be flushed by setting the IFF (Input FIFO Flush) bit of the JPEG_CR register.

Prior to flushing the FIFO:
- The interrupts for the input FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent unwanted DMA trigger.

The consequence of not flushing the FIFO at the end of the decoding process is that any remaining data is taken into the next JPEG decoding.

**Header parsing**

The header parsing can be activated setting the HDR bit of the JPEG_CONFR1 register.

The JPEG header parser supports all markers relevant to the JPEG baseline algorithm indicated in Annex B of the ISO/IEC 10918-1.

When parsing a supported marker, the JPEG header parser extracts the required parameters and stores them in shadow registers. At the end of the parsing the JPEG codec registers are updated.

If a DQT marker segment is located, quantization data associated with it is written into the quantization table memory.

If a DHT marker segment is located, the Huffman table data associated with it is converted into three different table formats (HuffMin, HuffBase and HuffSymb) and stored in their respective memories.

Once the parsing operation is completed, the HPDF (header parsing done flag) bit of the JPEG_SR register is set. An interrupt is generated if the EHPIE (end of header parsing interrupt enable) bit of the JPEG_CR register is set.

**JPEG decoding**

Once the JPEG header is parsed or JPEG codec registers and memories are properly programmed, the incoming data stream is decoded and the resulting MCUs are sent to the output FIFO.
When decoding two images successively, the START bit of the JPEG_CONFR0 register must be set again (even if already 1) after the header processing of the second image is completed.

**DMA generation for output FIFO**

DMA request is generated when the 32-byte output FIFO becomes at least half-full, that is, when there are at least 16 bytes of data.

A burst transfer is launched by the DMA to read 16 bytes of data.

Reads return 0 if the output FIFO is empty.

Once the decoding process is done, no extra bytes must remain in the output FIFO and no DMA request must be pending as the JPEG decoding generates blocks of 64 bytes.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed by setting the OFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the output FIFO must be disabled to prevent unwanted DMA request upon flushing the FIFO.

**Interrupt or DMA trigger generation for output FIFO**

The output FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO Threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. If the FIFO needs to be flushed, it must be done by software setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent unwanted DMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

### 34.3.4 JPEG encoding procedure

The JPEG codec can encode a JPEG stream as defined in the *ISO/IEC 10918-1* specification.

It can optionally generate the JPEG Header.

The JPEG codec is configured in encode mode resetting the DE bit (decode enable) of the JPEG_CONFR1 register.

The configuration used for encoding the JPEG must be loaded in the JPEG codec:

- JPEG codec configuration registers
- quantization tables
- Huffman tables
The JPEG codec is started setting the START bit of the JPEG_CONFR0 register.

Once the JPEG codec has been started, it request data for its input FIFO generating one of:
- DMA request
- DMA trigger
- interrupts

**DMA generation for input FIFO**

DMA request is generated when the 32-byte input FIFO becomes at least half-empty, that is, when there is free room for writing 16 bytes of data.

The DMA request generation is independent of the START bit of the JPEG_CONFR0 register. If the input FIFO can accept 16 bytes and the DMA for the input FIFO is enabled (setting the IDMAEN bit of the JPEG_CR register), a DMA request is generated regardless of the state of the JPEG codec kernel.

A burst transfer is launched by the DMA to write 16 bytes of data. Writes are ignored if the input FIFO is full.

At the end of the encoding process, extra bytes may remain in the input FIFO and/or a DMA request may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the input FIFO must be disabled to prevent unwanted DMA request upon flushing the FIFO.

The consequence of not flushing the FIFO at the end of the encoding process is that any remaining data is taken into the next JPEG encoding.

The DMA requests are no more generated once the EOCF flag of the JPEG_SR register is set.

**Interrupt or DMA trigger generation for input FIFO**

Input FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:
- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or DMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the encoding process, extra bytes may remain in the input FIFO and/or an interrupt request / DMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:
- The interrupts for the input FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent unwanted DMA trigger.

The consequence of not flushing the FIFO at the end of the encoding process is that any remaining data is taken into the next JPEG encoding.
JPEG encoding

Once the JPEG header generated, the incoming MCUs are encoded and the resulting data stream sent to the output FIFO.

DMA generation for output FIFO

DMA request is generated when the 32-byte output FIFO becomes at least half-full, that is, when there are at least 16 bytes of data.

A burst transfer is launched by the DMA to read 16 bytes of data.

Read returns 0 if the output FIFO is empty.

At the end of the encoding process, the last bytes may remain in the output FIFO as the stream padding may not be on 16 bytes.

These additional bytes must be managed by the CPU using the output FIFO not empty flag.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. The FIFO can be flushed by setting the OFF bit (output FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the input FIFO must be disabled to prevent unwanted DMA request upon flushing the FIFO.

Interrupt or DMA trigger generation for output FIFO

Output FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. The FIFO can be flushed by setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent unwanted DMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

The EOCF bit (end of conversion flag) of the JPEG_SR register can only be cleared when the output FIFO is empty.

Clearing either of the HDR bit (header processing) of the JPEG_CONFMR1 register and the JCEN bit (JPEG codec enable) of the JPEG_CR register is allowed only when the EOCF bit of the JPEG_SR register is cleared.
### 34.4 JPEG codec interrupts

An interrupt can be produced on the following events:
- input FIFO threshold reached
- input FIFO not full
- output FIFO threshold reached
- output FIFO not empty
- end of conversion
- header parsing done

Separate interrupt enable bits are available for flexibility.

#### Table 315. JPEG codec interrupt requests

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input FIFO threshold reached</td>
<td>IFTF</td>
<td>IFTIE</td>
</tr>
<tr>
<td>Input FIFO not full</td>
<td>IFNFF</td>
<td>IFNFIE</td>
</tr>
<tr>
<td>Output FIFO threshold reached</td>
<td>OFTF</td>
<td>OFTIE</td>
</tr>
<tr>
<td>Output FIFO not empty</td>
<td>OFNEF</td>
<td>OFNEIE</td>
</tr>
<tr>
<td>End of conversion</td>
<td>EOCF</td>
<td>EOCIE</td>
</tr>
<tr>
<td>Header parsing done</td>
<td>HPDF</td>
<td>HPDIE</td>
</tr>
</tbody>
</table>

### 34.5 JPEG codec registers

#### 34.5.1 JPEG codec control register (JPEG_CONFR0)

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

**Bit 0** START: Start

This bit start or stop the encoding or decoding process.

0: Stop/abort
1: Start

*Note: Reads always return 0.*
### 34.5.2 JPEG codec configuration register 1 (JPEG_CONFR1)

Address offset: 0x004  
Reset value: 0x0000 0000

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

**Bits 31:16 YSIZE[15:0]: Y Size**  
This field defines the number of lines in source image.

**Bits 15:9 Reserved, must be kept at reset value.**

**Bit 8 HDR: Header processing**  
This bit enables the header processing (generation/parsing).  
0: Disable  
1: Enable

**Bits 7:6 NS[1:0]: Number of components for scan**  
This field defines the number of components minus 1 for scan header marker segment.

**Bits 5:4 COLSPACE[1:0]: Color space**  
This field defines the number of quantization tables minus 1 to insert in the output stream.  
00: Grayscale (1 quantization table)  
01: YUV (2 quantization tables)  
10: RGB (3 quantization tables)  
11: CMYK (4 quantization tables)

**Bit 3 DE: Codec operation as coder or decoder**  
This bit selects the code or decode process  
0: Code  
1: Decode

**Bit 2 Reserved, must be kept at reset value.**

**Bits 1:0 NF[1:0]: Number of color components**  
This field defines the number of color components minus 1.  
00: Grayscale (1 color component)  
01: - (2 color components)  
10: YUV or RGB (3 color components)  
11: CMYK (4 color components)
34.5.3 JPEG codec configuration register 2 (JPEG_CONFR2)

Address offset: 0x008
Reset value: 0x0000 0000

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

Bits 25:0 **NMCU[25:0]**: Number of MCUs
For encoding: this field defines the number of MCU units minus 1 to encode.
For decoding: this field indicates the number of complete MCU units minus 1 to be decoded
(this field is updated after the JPEG header parsing). If the decoded image size has not a X or
Y size multiple of 8 or 16 (depending on the sub-sampling process), the resulting incomplete
or empty MCU must be added to this value to get the total number of MCUs generated.

34.5.4 JPEG codec configuration register 3 (JPEG_CONFR3)

Address offset: 0x00C
Reset value: 0x0000 0000

Bits 31:16 **XSIZE[15:0]**: X size
This field defines the number of pixels per line.

Bits 15:0 Reserved, must be kept at reset value.
### 34.5.5 JPEG codec configuration register x (JPEG_CONFRx)

Address offset: $0x000 + 0x4 \times x$, ($x = 4$ to $7$)

Reset value: $0x0000\ 0000$

<table>
<thead>
<tr>
<th>Register (Bit)</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>27 26 25 24</td>
<td>$\text{HSF}[3:0]$: Horizontal sampling factor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Horizontal sampling factor for component {x-4}.</td>
<td></td>
</tr>
<tr>
<td>23 22 21 20</td>
<td>$\text{VSF}[3:0]$: Vertical sampling factor</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vertical sampling factor for component {x-4}.</td>
<td></td>
</tr>
<tr>
<td>19 18 17 16</td>
<td>$\text{NB}[3:0]$: Number of blocks</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of data units minus 1 that belong to a particular color in the MCU.</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12</td>
<td>$\text{QT}[1:0]$: Quantization table</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selects quantization table used for component {x-4}.</td>
<td></td>
</tr>
<tr>
<td>11 10 9 8</td>
<td>00: Quantization table 0</td>
<td></td>
</tr>
<tr>
<td>11 10 9 8</td>
<td>01: Quantization table 1</td>
<td></td>
</tr>
<tr>
<td>11 10 9 8</td>
<td>10: Quantization table 2</td>
<td></td>
</tr>
<tr>
<td>11 10 9 8</td>
<td>11: Quantization table 3</td>
<td></td>
</tr>
<tr>
<td>7 6 5 4</td>
<td>$\text{HA}$: Huffman AC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selects the Huffman table for encoding AC coefficients.</td>
<td></td>
</tr>
<tr>
<td>3 2 1 0</td>
<td>0: Huffman AC table 0</td>
<td></td>
</tr>
<tr>
<td>3 2 1 0</td>
<td>1: Huffman AC table 1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$\text{HD}$: Huffman DC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Selects the Huffman table for encoding DC coefficients.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Huffman DC table 0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Huffman DC table 1</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 $\text{HSF}[3:0]$: Horizontal sampling factor
Horizontal sampling factor for component {x-4}.

Bits 11:8 $\text{VSF}[3:0]$: Vertical sampling factor
Vertical sampling factor for component {x-4}.

Bits 7:4 $\text{NB}[3:0]$: Number of blocks
Number of data units minus 1 that belong to a particular color in the MCU.

Bits 3:2 $\text{QT}[1:0]$: Quantization table
Selects quantization table used for component {x-4}.

00: Quantization table 0
01: Quantization table 1
10: Quantization table 2
11: Quantization table 3

Bit 1 $\text{HA}$: Huffman AC
Selects the Huffman table for encoding AC coefficients.

0: Huffman AC table 0
1: Huffman AC table 1

Bit 0 $\text{HD}$: Huffman DC
Selects the Huffman table for encoding DC coefficients.

0: Huffman DC table 0
1: Huffman DC table 1
34.5.6 JPEG control register (JPEG_CR)

Address offset: 0x030
Reset value: 0x0000 0000

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 OFF: Output FIFO flush
This bit flushes the output FIFO.
0: No effect
1: Output FIFO is flushed
Note: Reads always return 0.

Bit 13 IFF: Input FIFO flush
This bit flushes the input FIFO.
0: No effect
1: Input FIFO is flushed
Note: Reads always return 0.

Bit 12 ODMAEN: Output DMA enable
Enables DMA request generation for the output FIFO.
0: Disabled
1: Enabled

Bit 11 IDMAEN: Input DMA enable
Enables DMA request generation for the input FIFO.
0: Disabled
1: Enabled

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 HPDIE: Header parsing done interrupt enable
This bit enables interrupt generation upon the completion of the header parsing operation.
0: Disabled
1: Enabled

Bit 5 EOCIE: End of conversion interrupt enable
This bit enables interrupt generation at the end of conversion.
0: Disabled
1: Enabled

Bit 4 OFNEIE: Output FIFO not empty interrupt enable
This bit enables interrupt generation when the output FIFO is not empty.
0: Disabled
1: Enabled
Bit 3 **OFTIE**: Output FIFO threshold interrupt enable
   This bit enables interrupt generation when the output FIFO reaches a threshold.
   0: Disabled
   1: Enabled

Bit 2 **IFNFIE**: Input FIFO not full interrupt enable
   This bit enables interrupt generation when the input FIFO is not empty.
   0: Disabled
   1: Enabled

Bit 1 **IFTIE**: Input FIFO threshold interrupt enable
   This bit enables interrupt generation when the input FIFO reaches a threshold.
   0: Disabled
   1: Enabled

Bit 0 **JCEN**: JPEG core enable
   This bit enables the JPEG codec core.
   0: Disabled (internal registers are reset).
   1: Enabled (internal registers are accessible).

### 34.5.7 JPEG status register (JPEG_SR)

Address offset: 0x034

Reset value: 0x0000 0006

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 7 **COF**: Codec operation flag
   This bit flags code/decode operation in progress.
   0: Not in progress
   1: In progress

Bit 6 **HPDF**: Header parsing done flag
   In decode mode, this bit flags the completion of header parsing and updating internal registers.
   0: Not completed
   1: Completed

Bit 5 **EOCF**: End of conversion flag
   This bit flags the completion of encode/decode process and data transfer to the output FIFO.
   0: Not completed
   1: Completed
Bit 4 **OFNEF**: Output FIFO not empty flag
This bit flags that data is available in the output FIFO. This flag must not be considered when using DMA.
0: Empty (data not available)
1: Not empty (data available)

Bit 3 **OFTF**: Output FIFO threshold flag
This bit flags that the amount of data in the output FIFO reaches or exceeds a threshold. This flag must not be considered when using DMA.
0: Below threshold
1: At or above threshold

Bit 2 **IFNFF**: Input FIFO not full flag
This bit flags that the input FIFO is not full (data can be written). This flag must not be considered when using DMA.
0: Full
1: Not full

Bit 1 **IFTF**: Input FIFO threshold flag
This bit flags that the amount of data in the input FIFO is below a threshold. This flag must not be considered when using DMA.
0: At or above threshold
1: Below threshold.

Bit 0 Reserved, must be kept at reset value.

### 34.5.8 JPEG clear flag register (JPEG_CFR)
Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CHPDF**: Clear header parsing done flag
Writing 1 clears the HPDF bit of the JPEG_SR register.
0: No effect
1: Clear

Bit 5 **CEOCF**: Clear end of conversion flag
Writing 1 clears the ECF bit of the JPEG_SR register.
0: No effect
1: Clear

Bits 4:0 Reserved, must be kept at reset value.
34.5.9  JPEG data input register (JPEG_DIR)
Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 DATAIN[31:0]: Data input FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input FIFO data register</td>
</tr>
</tbody>
</table>

34.5.10  JPEG data output register (JPEG_DOR)
Address offset: 0x044
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 DATAOUT[31:0]: Data output FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output FIFO data register</td>
</tr>
</tbody>
</table>
34.5.11 JPEG quantization memory x (JPEG_QMEMx_y)  
Address offset: 0x050 + 0x40 * x + 0x4 * y, (x = 0 to 3; y = 0 to 15)  
Reset value: 0xXXXX XXXX  
Four quantization tables as specified by ISO documentation.  
For decoding with header parsing, no quantization table programming is required, the coefficients are directly written in the quantization memories by header parser.  
For decoding without header parsing or for encoding, the quantization table must be written by software in zig zag order.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x050 + 0x40 * x + 0x4 * y</th>
<th>(x = 0 to 3; y = 0 to 15)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value</td>
<td>0xXXXX XXXX</td>
<td></td>
</tr>
</tbody>
</table>

34.5.12 JPEG Huffman min (JPEG_HUFFMINx_y)  
Address offset: 0x150 + 0x10 * x + 0x4 * y, (x = 0 to 3; y = 0 to 2)  
Reset value: 0xXXXX XXXX  
This memory stores the minimum Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing.  
- DATA0: Min AC0 value  
- DATA1: Min DC0 value  
- DATA2: Min AC1 value  
- DATA3: Min DC1 value

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x150 + 0x10 * x + 0x4 * y</th>
<th>(x = 0 to 3; y = 0 to 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value</td>
<td>0xXXXX XXXX</td>
<td></td>
</tr>
</tbody>
</table>
34.5.13 JPEG Huffman min x (JPEG_HUFFMINx_y)

Address offset: 0x150 + 0x10 * x + 0x4 * y, (x = 0 to 3; y = 3)
Reset value: 0xXXXX XXXX

This memory stores the minimum Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing.

- DATA0: Min AC0 value
- DATA1: Min DC0 value
- DATA2: Min AC1 value
- DATA3: Min DC1 value

34.5.14 JPEG Huffman base (JPEG_HUFFBASEx)

Address offset: 0x190 + 0x4*x, (x = 0 to 31)
Reset value: 0xXXXX XXXX

This memory stores the base Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing:

- DATA0 to DATA15: Base AC0 value
- DATA16 to DATA31: Base DC0 value
- DATA32 to DATA47: Base AC1 value
- DATA48 to DATA63: Base DC1 value
34.5.15 JPEG Huffman symbol (JPEG_HUFFSYMBx)

Address offset: 0x210 + 0x4 * x, (x = 0 to 83)
Reset value: 0xXXXX XXXX

This memory stores the Huffman symbols used internally by the JPEG decoder. The memory content is written by hardware during the header parsing:
- DATA0 to DATA161: AC0 symbols
- DATA162 to DATA173: DC0 and DC1 symbols
- DATA174 to DATA335: AC1 symbols

| Bits 31:25 | Reserved, must be kept at reset value. |
| Bits 24:16 | DATA(2x+1)[8:0]: Data (2x+1) |
| Bits 15:9  | Reserved, must be kept at reset value. |
| Bits 8:0  | DATA(2x)[8:0]: Data (2x) |

| Bits 31:24 | DATA(4x+3)[7:0]: Data (4x+3) |
| Bits 23:16 | DATA(4x+2)[7:0]: Data (4x+2) |
| Bits 15:8  | DATA(4x+1)[7:0]: Data (4x+1) |
| Bits 7:0   | DATA(4x)[7:0]: Data (4x) |
### 34.5.16 JPEG DHT memory (JPEG_DHTMEMx)

Address offset: \(0x360 + 0x4 \times x\), \((x = 0 \text{ to } 102)\)

Reset value: \(0xXXXX \text{ XXXX}\)

For encoding process with header generation, this memory stores the DHT marker segment AC and DC Huffman tables in the ISO/IEC specification format:

- DATA0 to DATA27: DC Huffman table0
- DATA28 to DATA205: AC Huffman table0
- DATA206 to DATA233: DC Huffman table1
- DATA234 to DATA411: AC Huffman table1

### 34.5.17 JPEG Huffman encoder ACx (JPEG_HUFFENC_ACx_y)

Address offset: \(0x500 + 0x160 \times x + 0x4 \times y\), \((x = 0 \text{ to } 1; \ y = 0 \text{ to } 87)\)

Reset value: \(0xXXXX \text{ XXXX}\)

This memory defines the Huffman codes used during the encoding process of AC components.

Bits 31:24 \(DATA\{4x+3\}[7:0]\): Huffman table data \(4x+3\)

Huffman table data for DHT marker segment generation.

Bits 23:16 \(DATA\{4x+2\}[7:0]\): Huffman table data \(4x+2\)

Huffman table data for DHT marker segment generation.

Bits 15:8 \(DATA\{4x+1\}[7:0]\): Huffman table data \(4x+1\)

Huffman table data for DHT marker segment generation.

Bits 7:0 \(DATA\{4x\}[7:0]\): Huffman table data \(4x\)

Huffman table data for DHT marker segment generation.


34.5.18 JPEG Huffman encoder DCx (JPEG_HUFFENC_DCx_y)

Address offset: 0x7C0 + 0x20\*x + 0x4\*y, (x = 0 to 1; y = 0 to 7)

Reset value: 0xFFFF XXXX

This memory defines the Huffman codes used during the encoding process of DC components.

| Bits 31:28 | Reserved, must be kept at reset value. |
| Bits 27:24 | HLEN(2\*y+1)[3:0]: Huffman length (2\*y+1) |
  | Number of bits in the Huffman code HCODE(2\*y+1) minus 1. |
| Bits 23:16 | HCODE(2\*y+1)[7:0]: Huffman code (2\*y+1) |
  | 8 least significant bits of the Huffman code. |
  | If the Huffman code is less than 8 bits long, the unused bits must be 0. |
| Bits 15:12 | Reserved, must be kept at reset value. |
| Bits 11:8  | HLEN(2\*y)[3:0]: Huffman length (2\*y) |
  | Number of bits in the Huffman code HCODE(2\*y) minus 1. |
| Bits 7:0   | HCODE(2\*y)[7:0]: Huffman code (2\*y) |
  | 8 least significant bits of the Huffman code. |
  | If the Huffman code is less than 8 bits long, the unused bits must be 0. |

| Bits 31:28 | Reserved, must be kept at reset value. |
| Bits 27:24 | HLEN(2\*y+1)[3:0]: Huffman length (2\*y+1) |
  | Number of bits in the Huffman code HCODE(2\*y+1) minus 1. |
| Bits 23:16 | HCODE(2\*y+1)[7:0]: Huffman code (2\*y+1) |
  | 8 least significant bits of the Huffman code. |
  | If the Huffman code is less than 8 bits long, the unused bits must be 0. |
| Bits 15:12 | Reserved, must be kept at reset value. |
| Bits 11:8  | HLEN(2\*y)[3:0]: Huffman length (2\*y) |
  | Number of bits in the Huffman code HCODE(2\*y) minus 1. |
| Bits 7:0   | HCODE(2\*y)[7:0]: Huffman code (2\*y) |
  | 8 least significant bits of the Huffman code. |
  | If the Huffman code is less than 8 bits long, the unused bits must be 0. |
## 34.5.19 JPEG codec register map

The following table summarizes the JPEG codec registers. Refer to the register boundary addresses table for the JPEG codec register base address.

**Table 316. JPEG codec register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Reset value</th>
<th>Offset Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>JPEG_CONFR0</td>
<td>JPEG_CONFR1</td>
<td>Reset value</td>
<td>YSIZE[15:0]</td>
</tr>
<tr>
<td>0x004</td>
<td>JPEG_CONFR1</td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>JPEG_CONFR2</td>
<td>NMCU[25:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>JPEG_CONFR3</td>
<td>XSIZE[15:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>JPEG_CONFR4</td>
<td>HSF[3:0] VSF[3:0] NB[3:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>JPEG_CONFR5</td>
<td>HSF[3:0] VSF[3:0] NB[3:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>JPEG_CONFR7</td>
<td>HSF[3:0] VSF[3:0] NB[3:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x020-</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>JPEG_CR</td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>JPEG_SR</td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>JPEG_CFR</td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>JPEG_DIR</td>
<td>DATAIN[31:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>JPEG_DOR</td>
<td>DATAOUT[31:0]</td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x048-</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08C</td>
<td>JPEG_QMEM0</td>
<td>QCOEF[4<em>y+3][7:0] QCOEF[4</em>y+2][7:0] QCOEF[4<em>y+1][7:0] QCOEF[4</em>y][7:0]</td>
<td>Reset value</td>
<td></td>
</tr>
</tbody>
</table>

**Table 316. JPEG codec register map and reset values**
Refer to Section 2.3 for the register boundary addresses.
35 True random number generator (RNG)

35.1 Introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

35.2 RNG main features

- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if \( f_{\text{AHB}} < f_{\text{threshold}} \) (256 RNG clock cycles otherwise).
- It embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management.
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).
35.3 RNG functional description

35.3.1 RNG block diagram

*Figure 349* shows the RNG block diagram.

![Figure 349. RNG block diagram]

35.3.2 RNG internal signals

*Table 317* describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rng_it</td>
<td>Digital output</td>
<td>RNG global interrupt request</td>
</tr>
<tr>
<td>rng_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>rng_clk</td>
<td>Digital input</td>
<td>RNG dedicated clock, asynchronous to rng_hclk</td>
</tr>
<tr>
<td>rng_itamp_out</td>
<td>Digital output</td>
<td>RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.</td>
</tr>
</tbody>
</table>
35.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on Figure 350. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

![Figure 350. NIST SP800-90B entropy source model](Figure 350)

The components pictured above are detailed hereafter.

**Noise source**

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in Section 35.3.8: RNG low-power use. Multiple oscillators are also disabled for the configuration A (see Table 320: RNG configurations).
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

*Note:* In Section 35.6: RNG entropy source validation the recommended RNG clock frequencies and associated divider value are given.
Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in Section 35.6.2) a normalization debiasing is applied, that is half of the bits are taken from the sampled noise source, half of the bits are taken from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in Section 35.5: RNG processing time.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in Section 35.5: RNG processing time.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.
Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in Section 35.6.2).

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   – Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   – Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 times.
   – Known-answer tests, to verify the conditioning stage.

2. Continuous health tests, running indefinitely on the outputs of the noise source:
   – Repetition count test, similar to the one running in startup tests.
   – Adaptive proportion test, similar to the one running in startup tests.

3. Vendor specific continuous tests
   – Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   – Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.

4. On-demand test of digitized noise source (raw data)
   – Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in Section 35.3.4: RNG initialization). Other kinds of on-demand testing (software based) are not supported.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in Section 35.3.7: Error management.

Note: An interrupt can be generated when an error is detected. Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See Section 35.6: RNG entropy source validation for details.
35.3.4 RNG initialization

The RNG simplified state machine is pictured on Figure 351.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering startup behavior test on the raw data samples and known-answer tests.
3. When startup health tests are completed. During this time, three 128-bit noise source samples are used.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in Section 35.5: RNG processing time.

Figure 351. RNG initialization overview
Figure 351 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes.

Note: When the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

35.3.5 RNG operation

Normal operations

To run the RNG using interrupts, the following steps are recommended:

1. Consult Section 35.6: RNG entropy source validation and verify if a specific RNG configuration is required for the application.
   - If it is the case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
   - If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.
2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   - No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   - A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   - If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in Section 35.3.7 must be used.

To run the RNG in polling mode following steps are recommended:

1. Consult Section 35.6: RNG entropy source validation and verify if a specific RNG configuration is required for the application.
   - If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   - If it is not the case only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   - No error occurred (the SEIS and CEIS bits must be set to 0)
   - A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four
additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in Section 35.3.7 must be used.

**Note:** When data is not ready (DRDY = 0) RNG_DR returns zero. It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in Section 35.6: RNG entropy source validation. The gain in random number generation speed is summarized in Section 35.5: RNG processing time.

**Low-power operations**

If the power consumption is a concern to the application, low-power strategies can be used, as described in Section 35.3.8: RNG low-power use.

**Software post-processing**

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals. Built-in health check functions are described in Section 35.3.3: Random number generation.

### 35.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock. The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in Section 35.6: RNG entropy source validation.

**Note:** When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider must be higher than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).

See Section 35.3.1: RNG block diagram for details (AHB and RNG clock domains).

### 35.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

**Clock error detection**

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see Section 35.3.6: RNG clocking) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.
Note: The clock error has no impact on generated random numbers that is the application can still read the RNG_DR register.

**CEIS is set only when CECS is set to 1 by RNG.**

### Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred. If a value is available in the RNG_DR register, it must not be used as it may not have enough entropy.

The following sequence must be used to fully recover from a seed error:

1. **Software reset by writing CONDRST at 1 and at 0** (see bitfield description for details). This step is needed only if SECS is set. Indeed, when SEIS is set and SECS is cleared it means RNG performed the reset automatically (auto-reset). In this case application must clear the SEIS bit interrupt flag.
2. If SECS was set in step 1 (no auto-reset) wait for CONDRST to be cleared in the RNG_CR register, then confirm that SEIS is cleared in the RNG_SR register. Otherwise, just clear the SEIS bit in the RNG_SR register.
3. If SECS was set in step 1 (no auto-reset), wait for SECS to be cleared by RNG. The random number generation is now back to normal.

**Note:** After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the auto-reset is disabled. **CONDRST must be used in step 1.**

### RNG tamper errors

When an unexpected error is found by the RNG an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

### 35.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be reenabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done. Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in **Section 35.5: RNG processing time on page 1438.**

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates
all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

When the application wants to disable the RNG clock it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and disabling the RNG kernel clock using the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some marginal power consumption while RNGEN bit is set (RNG activated).

Note: The power modes where RNG is deactivated (that is retained or not available) can be found in the PWR section.

35.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see Section 35.3.7: Error management
- Clock error, see Section 35.3.7: Error management

Dedicated interrupt enable control bits are available as shown in Table 318.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNG</td>
<td>Data ready flag</td>
<td>DRDY</td>
<td>IE</td>
<td>None (automatic)</td>
</tr>
<tr>
<td></td>
<td>Seed error flag</td>
<td>SEIS</td>
<td>IE</td>
<td>Write CONDRST to 1 then to 0 unless ARDIS is cleared (see Section 35.3.7: Error management)</td>
</tr>
<tr>
<td></td>
<td>Clock error flag</td>
<td>CEIS</td>
<td>IE</td>
<td>Write 0 to CEIS</td>
</tr>
</tbody>
</table>

The user can enable or disable the above interrupt sources individually by changing the mask bits or the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.
35.5 RNG processing time

In recommended configuration A or C described in Table 320, the time between two sets of four 32-bit data is either:

- 203 x N AHB cycles if \( f_{\text{AHB}} < f_{\text{threshold}} \) (conditioning stage is limiting), or
- 128 x N RNG cycles \( f_{\text{AHB}} \geq f_{\text{threshold}} \) (noise source stage is limiting).

With \( f_{\text{threshold}} = 1.6 \times f_{\text{RNG}} \), for instance 77 MHz if \( f_{\text{RNG}} = 48 \text{ MHz} \). Value \( N \) is defined in Section 35.6: RNG entropy source validation.

Note: When CLKDIV is different from zero, \( f_{\text{RNG}} \) must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 203 AHB cycles if \( f_{\text{AHB}} < f_{\text{threshold}} \) or
- 32 RNG cycles \( f_{\text{AHB}} \geq f_{\text{threshold}} \)

with \( f_{\text{threshold}} = 6.5 \times f_{\text{RNG}} \).

Initialization time

More time is needed for the first set of random numbers after the device exits reset (see Section 1.3.4: RNG initialization). Table below gives details on how to compute the time spent in each initialization step.

<table>
<thead>
<tr>
<th>Initialization step</th>
<th>Configuration A or C, reset value</th>
<th>Configuration B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait for noise source, then startup health tests</td>
<td>Max( (wait_noise(1) + 11 + 1024 x CLKDIV) RNG_cycles, 13 x 203 AHB_cycles)</td>
<td>Max(16 + 1035 RNG_cycles, 13 x 203 AHB_cycles)</td>
</tr>
<tr>
<td>Conditioning keys initialization</td>
<td>Max((1+2xN) x 128 x CLKDIV) RNG_cycles + 203 AHB_cycles, (128 x CLKDIV) RNG_cycles + (2xN+2) x 203 AHB_cycles)</td>
<td>Max (3 x 32 RNG_cycles + 203 AHB_cycles, 32 RNG_cycles + 4 x 203 AHB_cycles)</td>
</tr>
</tbody>
</table>

1. 192 RNG_cycles (configuration A or C), 16 RNG_cycles (reset value)

As an example, if AHB clock= 250 MHz and RNG clock= 48 MHz then the initialization time is around 40 \( \mu \text{s} \) in the configuration C (CLKDIV=1, N=2) and 26 \( \mu \text{s} \) in the configuration B.

35.6 RNG entropy source validation

35.6.1 Introduction

In order to assess the amount of entropy available from the RNG, STMicroelectronics has tested the peripheral using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.
35.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk= 48 MHz
- RNG configurations are described in Table 320: RNG configurations. Note that only configuration A can be certified NIST SP800-90B. Refer to Table 321: Configuration selection to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 320. RNG configurations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Refer to NIST compliant RNG configuration table in AN4230 available from <a href="http://www.st.com">www.st.com</a>. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on <a href="https://csrc.nist.gov/projects/cryptographic-module-validation-program">https://csrc.nist.gov/projects/cryptographic-module-validation-program</a>.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0x18</td>
<td>0x0</td>
<td>0x0</td>
<td>0x0</td>
<td>0</td>
<td>1</td>
<td>0x0000 AAC7(4)</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0x0F</td>
<td>0x0</td>
<td>0x0D</td>
<td>0x0</td>
<td>0</td>
<td>2</td>
<td>default</td>
</tr>
</tbody>
</table>

1. 0x1 value is recommended instead of 0x0 for RNG_CONFIG2[2:0], when RNG power consumption is critical. See the end of Section 35.3.8: RNG low-power use for details.
2. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4
3. The noise source sampling must be 48 MHz or less. Hence, if the RNG clock is different from 48 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in Section 35.7.1 for details.
4. This value can be fixed in the RNG driver (it doesn’t depend on the STM32 family).

Table 321. Configuration selection

<table>
<thead>
<tr>
<th>Section criteria</th>
<th>Config A</th>
<th>Config B</th>
<th>Config C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Suitable to generate NIST compliant cryptographic keys</td>
<td>Yes</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Entropy(1)</td>
<td>Certified</td>
<td>Good</td>
<td>Very good</td>
</tr>
<tr>
<td>Speed(2)</td>
<td>Baseline</td>
<td>Faster</td>
<td>Baseline</td>
</tr>
</tbody>
</table>

1. For configurations B and C entropy is verified using AIS-31 test suite (T0 to T8).
2. When speed is not enough for application a NIST compliant DRBG can be used to increase throughput.

35.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

35.7.1 RNG control register (RNG_CR)

Address offset: 0x000
Reset value: 0x0080 0D00

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>CONFIGLOCK: RNG Config lock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Writes to the RNG_NSCR, RNG_HT CR and RNG_CR configuration bits [29:4] are allowed.</td>
</tr>
<tr>
<td>1</td>
<td>Writes to the RNG_NSCR, RNG_HT CR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.</td>
</tr>
<tr>
<td></td>
<td>Once set, this bit can only be cleared when RNG is reset (set once bit).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>CONDRST: Conditioning soft reset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR, RNG_NSCR and RNG_HT CR are not changed by CONDRST.</td>
</tr>
<tr>
<td></td>
<td>This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.</td>
</tr>
<tr>
<td></td>
<td>When CONDRST is set to 0 by the software, its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.</td>
</tr>
</tbody>
</table>

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:20 RNG_CONFIG1[5:0]: RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in Section 35.6: RNG entropy source validation.

Writing any bit of RNG_CONFIG1 is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 19:16 CLKDIV[3:0]: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.
0x1: two RNG clock cycles per internal RNG clock.
0x2: \(2^2 = 4\) RNG clock cycles per internal RNG clock.
...
0xF: \(2^{15}\) RNG clock cycles per internal clock (for example, an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock).

Writing these bits is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.
Bits 15:13 **RNG_CONFIG2[2:0]**: RNG configuration 2
Reserved to the RNG configuration (bitfield 2). Bit 13 can be set when RNG power consumption is critical. See [Section 35.3.8: RNG low-power use](#). Refer to the RNG_CONFIG1 bitfield for details.

**Bit 12 NISTC**: NIST custom
0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.
1: Custom values for NIST compliant RNG. See [Section 35.6: RNG entropy source validation](#) for recommended configuration.
Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 11:8 **RNG_CONFIG3[3:0]**: RNG configuration 3
Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details. If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

**Bit 7 ARDIS**: Auto reset disable
Set this bit to deactivate the auto-reset feature.
0: Auto-reset enabled
1: Auto-reset disabled
Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in [Section 35.3.7: Error management](#).
Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

**Bit 6 Reserved, must be kept at reset value.**

**Bit 5 CED**: Clock error detection
0: Clock error detection enabled
1: Clock error detection is disabled
The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.
Writing this bit is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

**Bit 4 Reserved, must be kept at reset value.**

**Bit 3 IE**: Interrupt enable
0: RNG interrupt is disabled
1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS is set in the RNG_SR register.

**Bit 2 RNGEN**: True random number generator enable
0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.
1: True random number generator is enabled.

**Bits 1:0** Reserved, must be kept at reset value.
35.7.2  RNG status register (RNG_SR)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:7</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

**Bit 6  SEIS:** Seed error interrupt status
This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

- 0: No faulty sequence detected
- 1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

**Bit 5  CEIS:** Clock error interrupt status
This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

- 0: The RNG clock is correct (fRNGCLK > fHCLK/32)
- 1: The RNG clock before the internal divider is detected too slow (fRNGCLK < fHCLK/32)

An interrupt is pending if IE = 1 in the RNG_CR register.

**Bits 4:3** Reserved, must be kept at reset value.

**Bit 2  SECS:** Seed error current status
0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.
1: At least one of the following faulty sequences has been detected:
- Runtime repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Startup or continuous adaptive proportion test on noise source failed.
- Startup post-processing/conditioning sanity check failed.

**Bit 1  CECS:** Clock error current status
0: The RNG clock is correct (fRNGCLK > fHCLK/32). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.
1: The RNG clock is too slow (fRNGCLK < fHCLK/32).

*Note:* CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

**Bit 0  DRDY:** Data ready
0: The RNG_DR register is not yet valid, no random data is available.
1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

*Note:* The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.
35.7.3 **RNG data register (RNG_DR)**

Address offset: 0x008
Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **RNDATA[31:0]**: Random data
32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.
When DRDY is set, it is recommended to always verify that RNG_DR is different from zero.
The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

35.7.4 **RNG noise source control register (RNG_NSCR)**

Address offset: 0x00C
Reset value: 0x0003 FFFF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 17:15 **EN_OSC6[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 6. The bitfield has no effect otherwise.

Bits 14:12 **EN_OSC5[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 5. The bitfield has no effect otherwise.
Bits 11:9 **EN_OSC4[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 4. The bitfield has no effect otherwise.

Bits 8:6 **EN_OSC3[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**
When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

### 35.7.5 RNG health test control register (RNG_HTCR)

**Address offset:** 0x010

**Reset value:** 0x0000 72AC

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**HTCFG[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**HTCFG[15:0]**

| Bits 31:0 **HTCFG[31:0]:** health test configuration
This configuration is used by RNG to configure the health tests. See **Section 35.6: RNG entropy source validation** for the recommended value.

**Note:** The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.
### 35.7.6 RNG register map

Table 322. RNG register map and reset map

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | RNG_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | RNG_SR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | RNG_DR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x00C  | RNG_NSCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  |
| 0x010  | RNG_HTCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3: Memory organization for the register boundary addresses.
36 Secure AES coprocessor (SAES)

36.1 Introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/decryption (wrapped-key mode) and key sharing with faster CRYP peripheral (shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK, application hardware key AHK, and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the CRYP peripheral.

36.2 SAES main features

- Compliant with NIST FIPS publication 197 “Advanced encryption standard (AES)” (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), certified SESIP and PSA security assurance level 3
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster CRYP peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together
  - Optional 128-bit or 256-bit hardware loading of non-volatile application hardware keys (AHK), secret to software and stored in the embedded flash memory

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK, application hardware key AHK, and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the CRYP peripheral.
36.3 SAES implementation

The devices have one SAES peripheral, connected with TAMP backup registers (BHK - boot hardware key) and with the Flash memory interface (AHK - application hardware key), and implemented as per the following table. It shares the key with the CRYP peripheral. For comparison, the CRYP peripheral is also included in the table.

Table 323. CRYP versus SAES features

<table>
<thead>
<tr>
<th>Modes or features(1)</th>
<th>CRYP</th>
<th>SAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB, CBC chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CTR, CCM, GCM chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AES 128-bit ECB encryption in cycles</td>
<td>14</td>
<td>480</td>
</tr>
<tr>
<td>DHUK and BHK key selection</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Resistance to side-channel attacks</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Shared key between SAES and CRYP</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Key sizes in bits</td>
<td>128, 192, 256</td>
<td>128, 256</td>
</tr>
</tbody>
</table>

1. X = supported.

36.4 SAES functional description

36.4.1 SAES block diagram

_Figure 352_ shows the block diagram of SAES.
36.4.2 SAES internal signals

*Table 324* describes the user relevant internal signals interfacing the SAES peripheral.

**Table 324. SAES internal input/output signals**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>saes_hclk</td>
<td>Input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>saes_ker_ck</td>
<td>Input</td>
<td>SAES kernel clock.</td>
</tr>
<tr>
<td>saes_it</td>
<td>Output</td>
<td>SAES interrupt request</td>
</tr>
<tr>
<td>saes_in_dma</td>
<td>Input/Output</td>
<td>SAES incoming data DMA single request/acknowledge</td>
</tr>
<tr>
<td>saes_out_dma</td>
<td>Input/Output</td>
<td>SAES processed data DMA single request/acknowledge</td>
</tr>
<tr>
<td>saes_itamp_out</td>
<td>Output</td>
<td>Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.</td>
</tr>
<tr>
<td>RHUK</td>
<td>Input</td>
<td>256-bit root hardware unique key (non-volatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)</td>
</tr>
<tr>
<td>BHK(1)</td>
<td>Input</td>
<td>256-bit boot hardware key (BHK) stored in tamper-resistant backup registers and written during boot. Once written, this key cannot be read nor written by any application until the next product reset.</td>
</tr>
</tbody>
</table>
36.4.3 **SAES reset and clocks**

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

36.4.4 **SAES symmetric cipher implementation**

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in Table 325. Those functions can be certified NIST PUB 140-3.

**Table 325. SAES approved symmetric key functions**

<table>
<thead>
<tr>
<th>Operations</th>
<th>Algorithm</th>
<th>Specification</th>
<th>Key bit lengths</th>
<th>Chaining modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption, decryption</td>
<td>AES</td>
<td>FIPS PUB 197</td>
<td>128, 256</td>
<td>ECB, CBC, CTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NIST SP800-38A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticated encryption or decryption</td>
<td>AES</td>
<td>NIST SP800-38C</td>
<td></td>
<td>GCM, CCM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NIST SP800-38D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cipher-based message</td>
<td></td>
<td>NIST SP800-38D</td>
<td></td>
<td>GMAC</td>
</tr>
<tr>
<td>authentication code</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).

It is possible to suspend then resume any SAES processing, following the sequence described in Section 36.4.8.
36.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

**Figure 353. Encryption/decryption typical usage**

![Flowchart diagram](image)

**Initialization**

The SAES peripheral is initialized according to the chaining mode. Refer to Section 36.4.9: SAES basic chaining modes (ECB, CBC) and Section 36.4.10: SAES counter (CTR) mode for details.

**Data append**

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to Section 36.4.7: SAES ciphertext stealing and data padding if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

**Appending data using the CPU in polling mode**

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using
NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor. NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in Section 36.6: SAES DMA requests. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer must not include the last four-word block if padded with zeros. The sequence described in Appendix data using the CPU in polling mode must be used instead for this last block, because the
NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

36.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 354. Typical operation with authentication

Section 36.4.11: SAES Galois/counter mode (GCM) and Section 36.4.13: SAES counter with CBC-MAC (CCM) describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in Section 36.4.12: SAES Galois message authentication code (GMAC).

36.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, Recommendation for Block
Ciphertext Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since SAES does not implement such techniques, the application must complete the last block of input data using data from the second last block.

**Note:** Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the last block, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

### 36.4.8 SAES suspend and resume operations

A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode. Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block. *Figure 355* gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

*Figure 355. Example of suspend mode management*

A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

### 36.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.
Figure 356 and Figure 357 describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

**Figure 356. ECB encryption**

![Diagram of ECB encryption](Image1)

**Figure 357. ECB decryption**

![Diagram of ECB decryption](Image2)

In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.
**Figure 358** and **Figure 359** describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*
ECB and CBC encryption process

This process is described in Section 36.4.5, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in encryption mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 36.4.14 (wrapped keys) and Section 36.4.15 (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 36.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in Section 36.4.7.
   b) Append the cleartext block into SAES as described in Section 36.4.5, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step b) until the third-last plaintext block is encrypted. For the last two blocks, follow the steps a) and b).
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in Section 36.4.5, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the key derivation mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 36.4.14 (wrapped keys) and Section 36.4.15 (shared keys).
4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 36.4.17: SAES key registers for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.

7. Wait until the CCF flag in the SAES_ISR register is set.

8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.

9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in decryption mode (write MODE[1:0] with 0x2). Do not change other parameters.

10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.

11. Enable the SAES peripheral, by setting EN.

12. Append encrypted data:
   a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in Section 36.4.7.
   b) Append the ciphertext block into SAES as described in Section 36.4.5, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
   c) Repeat the step b) until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps a) and b).

13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

**Suspend/resume operations in ECB/CBC modes**

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

**To suspend the processing of a message**, proceed as follows:

1. If DMA is used, stop the SAES DMA transfers to the input FIFO, by clearing DMAINEN. If DMA is not used, wait until the CCF flag in the SAES_ISR register is set (computation completed).

2. If DMA is not used, read four times the SAES_DOUTR register to save the last processed block. If DMA is used, wait until the CCF flag is set in the SAES_ISR register then stop the DMA transfers from the output FIFO, by clearing DMAOUTEN.

3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).

5. Disable the SAES peripheral, by clearing EN.

6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

7. If DMA is used, save the DMA controller status (pointers for SAES input and output data transfers, number of remaining bytes, and so on).

**To resume the processing of a message**, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the remaining input FIFO and output FIFO transfers.

2. Disable the SAES peripheral, by clearing EN.

3. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to Section 36.4.17: SAES key registers for details.

4. Prepare the decryption key, as described in ECB/CBC decryption process (only required for ECB or CBC decryption).
5. Restore the SAES_IvRx registers, using the saved configuration (only required in CBC mode).
6. Enable the SAES peripheral, by setting EN.
7. If DMA is used, enable SAES DMA transfers, by setting DMAINEN and DMAOUTEN.

Note: It is not required to save the key registers as the application knows the original key.

36.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in Figure 360.

Figure 360. Message construction in CTR mode

The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - Initialization vector (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - Counter: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.

- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

CTR encryption and decryption

Figure 361 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.
Initialization vectors in SAES must be initialized as shown in Table 326.

Table 326. Counter mode initialization vector definition

<table>
<thead>
<tr>
<th>SAES_IVR3[31:0]</th>
<th>SAES_IVR2[31:0]</th>
<th>SAES_IVR1[31:0]</th>
<th>SAES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit counter</td>
<td>32-bit counter</td>
<td>32-bit counter</td>
<td>32-bit counter</td>
</tr>
<tr>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
<td>0x0001</td>
</tr>
</tbody>
</table>

CTR encryption and decryption process

This process is described in Section 36.4.5, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 36.4.14 (wrapped keys) and Section 36.4.15 (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to Table 326.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 36.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in
      the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 36.4.5, then read the
      SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step b) until the second-last block is processed. For the last block of
      plaintext (encryption only), follow the steps a) and b). For the last block, discard
      the bits that are not part of the message when the last block is smaller than 16
      bytes.

9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority
message, then resume the interrupted message. Detailed CBC suspend and resume
sequence is described in Section 36.4.9: SAES basic chaining modes (ECB, CBC).

Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

36.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext
message into the corresponding ciphertext and tag (also known as message authentication
code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a
fixed finite field for computing the message authentication code. The following figure shows
a typical message construction in GCM mode.

Figure 362. Message construction in GCM

![Figure 362. Message construction in GCM](MSv42157V1)
The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.

- **Authenticated header AAD** (also known as additional authentication data) has a known length \( \text{Len}(A) \) that may be a non-multiple of 16 bytes, and must not exceed \( 2^{64} - 1 \) bits. This part of the message is only authenticated, not encrypted.

- **Plaintext message** \( P \) is both authenticated and encrypted as ciphertext \( C \), with a known length \( \text{Len}(P) \) that may be non-multiple of 16 bytes, and cannot exceed \( 2^{32} - 2 \) 16-byte blocks.

- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in Table 328.

The GCM standard specifies that ciphertext \( C \) has the same bit length as the plaintext \( P \).

When a part of the message (AAD or \( P \)) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*. 
Figure 363 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in Table 327.

<table>
<thead>
<tr>
<th>Table 327. Initialization of IV registers in GCM mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAES_IVR3[31:0]</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>32-bit counter = 0x0002</td>
</tr>
</tbody>
</table>

The last block of a GCM message contains the AAD header length and the payload length information, as shown in Table 328.

<table>
<thead>
<tr>
<th>Table 328. GCM last block definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word order to SAES_DINR</td>
</tr>
<tr>
<td>-------------------------</td>
</tr>
</tbody>
</table>
GCM encryption and decryption process

This process is described in Section 36.4.6, with the following sequence of events:

**GCM initialize**

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 36.4.14 (wrapped keys) and Section 36.4.15 (shared keys).
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to Table 327.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 36.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

**GCM header phase**

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in Section 36.4.5.
    c) Repeat the step b) until the second-last AAD data block is processed. For the last block, follow the steps a) and b).

*Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0. No data are read during header phase.*

**GCM payload phase**

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
   a) If it is the last block and the message in the block is smaller than 16 bytes, pad the
      remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 36.4.5, then read the
      SAES_DOUTR register four times to save the resulting block
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last
      block of ciphertext is decrypted. For the last block of plaintext (encryption only),
      follow the steps a) and b). For the last block, discard the bits that are not part of
      the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.

15. Select the GCM final phase, by writing GCMFH[1:0] with 0x3. Do not change the other
    configurations written during GCM initialization.

16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit
    and payload bit lengths, as shown in Table 328.

17. Wait until the CCF flag in the SAES_ISR register is set.

18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.

19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption,
    compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping
      is applied to tag data read from SAES_DOUTR.

When transitioning from the header or the payload phase to the final phase, the SAES
    peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

Suspend/resume operations are not supported in GCM mode.

36.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext,
    generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext
    authenticated data (that is, only header, no payload). The following figure shows typical
    message construction for GMAC.
Figure 364. Message construction in GMAC mode

For more details, refer to NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC*. Figure 365 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 365. GMAC authentication mode

The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

**Suspend/resume operations in GMAC**

Suspend/resume operations are not supported in GMAC mode.
36.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

*Note:* NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

**Figure 366. Message construction in CCM mode**

The structure of the message is:

- **16-byte first authentication block (B0),** composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values t (MAC length expressed in bytes) and Q (plaintext length such that Len(P) < 2^8Q bytes). The counter blocks range associated to Q is equal to 2^8Q-4, that is, if the maximum value of Q is 8, the counter blocks used in cipher must be on 60 bits.

- **16-byte blocks (B) associated to the associated data (A).**
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see Figure 366). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If 0 < a < 2^16 - 2^8, then it is encoded as [a]16, that is, on two bytes.
  - If 2^16 - 2^8 < a < 2^32, then it is encoded as 0xff || 0xfe || [a]32, that is, on six bytes.
  - If 2^32 < a < 2^64, then it is encoded as 0xff || 0xff || [a]64, that is, on ten bytes.
• **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see Figure 366).

• **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

**Note:** CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

- N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)
- A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)
- P: 20212223 (Len(P) = 32 bits or 4 bytes)
- T: 6084341B (Len(T) = 32 bits or t = 4)
- B0: 4F101112 13141516 00000000 00000004
- B1: 00080001 02030405 06070000 00000000
- B2: 20212223 00000000 00000000 00000000
- CTR0: 0710111213 141516 00000000 00000000
- CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, *Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*. 
Figure 367 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 367. CCM mode authenticated encryption

The first block of a CCM message (B0) must be prepared by the application as defined in Table 329.

Table 329. Initialization of IV registers in CCM mode

<table>
<thead>
<tr>
<th>SAES_IVR3[31:0]</th>
<th>SAES_IVR2[31:0]</th>
<th>SAES_IVR1[31:0]</th>
<th>SAES_IVR0[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0[127:96]</td>
<td>B0[95:64]</td>
<td>B0[63:32]</td>
<td>B0[31:0]</td>
</tr>
</tbody>
</table>

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.
CCM encryption and decryption process

This process is described in Section 36.4.6, with the following sequence of events:

CCM initialize
1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to Section 36.4.14 (wrapped keys) and Section 36.4.15 (shared keys).
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to Table 329.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to Section 36.4.17: SAES key registers for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase
10. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
   c) Enable the SAES peripheral, by setting EN.
11. Append header data:
   a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 36.4.5.
   c) Repeat the step b) until the second-last associated data block is processed. For the last block, follow the steps a) and b).

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0
No data are read during the header phase.
CCM payload phase

12. Initialize payload phase:
   a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
   b) If the header phase is skipped, enable the SAES peripheral, by setting EN.

13. Append payload data:
   a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in Section 36.4.5, then read the SAES_DOUTR register four times to save the resulting block.
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).
Remove LSB_{Len(T)} encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.

15. Wait until CCF flag in the SAES_ISR register is set.

16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.

17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.
When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the SAES DMA transfers to the input FIFO, by clearing DMAINEN. If DMA is not used, wait until the CCF flag of the SAES_ISR register is set (computation completed).

2. In the payload phase, if DMA is not used, read four times the SAES_DOUTR register to save the last-processed block. If DMA is used, wait until the CCF flag in the SAES_ISR register is set, then stop the DMA transfers from the output FIFO, by clearing DMAOUTEN.

3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.

4. Save the SAES_SUSPRx registers in the memory.

5. Save the IV registers as they are altered during the data processing.

6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
8. If DMA is used, save the DMA controller status (pointer for SAES input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for SAES output data transfers.

To resume the processing of a message, proceed as follows:
1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the SAES peripheral, by clearing EN.
3. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
4. Restore SAES_IVRx registers using the saved configuration.
5. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to Section 36.4.17: SAES key registers for details.
6. Enable the SAES peripheral, by setting EN.
7. If DMA is used, enable SAES DMA requests, by setting DMAINEN (and DMAOUTEN if in payload phase).

36.4.14 SAES operation with wrapped keys

SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK or AHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 368 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1, 0x5, or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral. AHK can also be selected, by writing KEYSEL[2:0] with 0x3.
Figure 368. Operation with wrapped keys for SAES in ECB and CBC modes

Wrapped-key mode (KMOD = 01)

Normal-key mode (KMOD = 00)

<table>
<thead>
<tr>
<th>Step 1: provision</th>
<th>Step 2: load</th>
<th>Step 3: use</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIN</td>
<td>DIN</td>
<td>DIN</td>
</tr>
<tr>
<td>SAES enc</td>
<td>SAES dec</td>
<td>SAES enc/dec</td>
</tr>
<tr>
<td>DOUT</td>
<td>DOUT</td>
<td>DOUT</td>
</tr>
<tr>
<td>wrapped (encrypted) key</td>
<td>unwrapped (decrypted) key</td>
<td></td>
</tr>
</tbody>
</table>

Note: DHUK value depends on privilege, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in encryption mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1, 0x5 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to Section 36.4.17 for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see Table 331 on page 1479).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF bit, by setting the CCF bit in SAES_ICR register.
11. Repeat steps 8. to 10. if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.
Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in key derivation mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral.
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the decryption mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see Table 331 on page 1479).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode).

Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

When KEYSEL[2:0] is at 0x1 (DHUK), 0x5 (DHUK XOR AHK), or 0x4 (DHUK XOR BHK), the application software must use the same privilege, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 369 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1, 0x5 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMPER peripheral.
The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1, 0x5 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to Section 36.4.17 for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps 8. and 9. if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode).
Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_CR register.

36.4.15 SAES operation with shared keys

SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK or AHK. With this feature, the application software can make the AES keys available to the CRYP peripheral, without exposing them in cleartext (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Figure 370 summarizes how to wrap or unwrap keys to share with CRYP peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1, 0x5, or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral. AHK can also be selected, by writing KEYSEL[2:0] with 0x3.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.
Key unwrapping for CRYP peripheral (shared key)

Each time SAES needs to share a key with the CRYP peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by CRYP. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in Figure 370. See Key unwrapping for SAES for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the CRYP peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the CRYP peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the CRYP key share target peripheral is described in AES key sharing with secure AES co-processor of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the CRYP peripheral, and KEYSIZE value of CRYP and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

36.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See Table 330 “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to
the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in Table 330 and Figure 371.

Table 330. AES data swapping example

<table>
<thead>
<tr>
<th>DATATYPE[1:0]</th>
<th>Swapping performed</th>
<th>Data block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>System memory data (big-endian)</td>
</tr>
<tr>
<td>0x0</td>
<td>No swapping</td>
<td>Block[127..64]: 0x04EEF672 2E04CE96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[63..0]: 0x4E6F7720 69732074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[127..96]: 0x04EEF672</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[95..64]: 0x2E04CE96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x8, word[63..32]: 0x4E6F7720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0xC, word[31..0]: 0x69732074</td>
</tr>
<tr>
<td>0x1</td>
<td>Half-word (16-bit) swapping</td>
<td>Block[63..0]: 0x4E6F7720 69732074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x7720 4E6F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2074 6973</td>
</tr>
<tr>
<td>0x2</td>
<td>Byte (8-bit) swapping</td>
<td>Block[63..0]: 0x4E 6F 77 20 69 73 20 74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x2077 6F4E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x7420 7369</td>
</tr>
<tr>
<td>0x3</td>
<td>Bit swapping</td>
<td>Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0110 0010 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[31..0]: 0x69732074 0110 1001 0111 0010 0000 0111 0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110</td>
</tr>
</tbody>
</table>
**Figure 371. 128-bit block construction according to the data type**

**DATATYPE[1:0] = 00: no swapping**

**DATATYPE[1:0] = 01: 16-bit (half-word) swapping**

**DATATYPE[1:0] = 10: 8-bit (byte) swapping**

**DATATYPE[1:0] = 11: bit swapping**

Legend:  
- $D_x$: input/output data bit 'x'  
- Data swap  
- Zero padding (example)  
- AES core input/output data

---

**Data padding**

*Figure 371* also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with $\text{DATATYPE}[1:0] = 0x0$
- 48 message bits, with $\text{DATATYPE}[1:0] = 0x1$
- 56 message bits, with $\text{DATATYPE}[1:0] = 0x2$
- 34 message bits, with $\text{DATATYPE}[1:0] = 0x3$
### 36.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on Table 331. Reads are not allowed for security reason.

**Note:** In memory and in SAES key registers, keys are stored in little-endian format, with most significant byte on the highest address.

#### Table 331. Key endianness in SAES_KEYRx registers (128/256-bit keys)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>KEY[127:96]</td>
<td>KEY[95:64]</td>
<td>KEY[63:32]</td>
<td>KEY[31:0]</td>
</tr>
</tbody>
</table>

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

**Note:** KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see Section 36.4.19).

#### Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a non-volatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information.

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to Table 331.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to Table 331.
With KEYSEL[2:0] at 0x3, the application hardware key (AHK), stored in the embedded Flash memory, is entirely transferred into key registers upon performing a successful option byte key read.

With KEYSEL[2:0] at 0x5, the XOR combination of DHUK and AHK is entirely transferred into key registers upon performing a successful option byte read.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK, AHK, or BHK if KEYVALID is set. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transferred to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable.

If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see Section 36.4.19).

Note: DHUK, AHK, BHK and their XOR combination are not readable by software.

### 36.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in Table 332. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

<table>
<thead>
<tr>
<th>SAES_IVR3[31:0]</th>
<th>SAES_IVR2[31:0]</th>
<th>SAES_IVR1[31:0]</th>
<th>SAES_IVR0[31:0]</th>
</tr>
</thead>
</table>

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

### 36.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

#### Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to Section 36.5: SAES interrupts.
The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

**Write error flag (WRERRF)**

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

*Note:* Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to Section 36.5: SAES interrupts.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

**Key error interrupt flag (KEIF)**

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See Section 36.4.17: SAES key registers for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in CRYP peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to CRYP peripheral. See Section 36.4.15: SAES operation with shared keys for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK, AHK, or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK), 0x3 (AHK), 0x5 (DHUK XOR AHK), or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to Section 36.5: SAES interrupts.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both CRYP and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

*Note:* For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

**RNG error interrupt flag (RNGEIF)**

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the
RNGEIE bit of the SAES_IER register is set. For more details, refer to Section 36.5: SAES interrupts.

Upon an RNG error:

- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

**Note:** To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

**About DPA errors**

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

### 36.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See Section 36.4.19: SAES error management for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

**Table 333** gives a summary of the available features.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAES</td>
<td>computation completed flag</td>
<td>CCF</td>
<td>CCFIE</td>
<td>set CCF(1)</td>
</tr>
<tr>
<td></td>
<td>read error flag</td>
<td>RDERRF(2)</td>
<td>RWEIE</td>
<td>set RWEIF(1)</td>
</tr>
<tr>
<td></td>
<td>write error flag</td>
<td>WRERRF(2)</td>
<td>KEIE</td>
<td>set KEIF(1)</td>
</tr>
<tr>
<td></td>
<td>key error flag</td>
<td>KEIF</td>
<td>KEIE</td>
<td>set KEIF(1)</td>
</tr>
<tr>
<td></td>
<td>RNG error flag</td>
<td>RNGEIF</td>
<td>RNGEIF</td>
<td>set RNGEIF(1)</td>
</tr>
</tbody>
</table>

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

36.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Detailed usage of DMA with SAES can be found in Appending data using DMA subsection of Section 36.4.5: SAES encryption or decryption typical usage.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 36.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

36.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Clock cycles(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Encryption or decryption(2)</td>
<td>ECB, CBC, CTR</td>
<td>480</td>
</tr>
<tr>
<td></td>
<td>Key preparation</td>
<td>-</td>
<td>145</td>
</tr>
</tbody>
</table>
Table 334. Processing latency for ECB, CBC and CTR (continued)

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Clock cycles(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>256-bit</td>
<td>Encryption or decryption(2)</td>
<td>ECB, CBC, CTR</td>
<td>680</td>
</tr>
<tr>
<td></td>
<td>Key preparation</td>
<td>-</td>
<td>230</td>
</tr>
</tbody>
</table>

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 335. Processing latency for GCM and CCM (in SAES kernel clock cycles)

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Initialization phase</th>
<th>Header phase(1)</th>
<th>Payload phase(1)</th>
<th>Final phase(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>128-bit</td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td>490</td>
<td>72(2)</td>
<td>480(3)</td>
<td>490</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>490</td>
<td>490</td>
<td>800</td>
<td>490</td>
</tr>
<tr>
<td>256-bit</td>
<td>Mode 1: Encryption/Mode 3: Decryption</td>
<td>GCM</td>
<td>650</td>
<td>72(2)</td>
<td>690(3)</td>
<td>650</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCM</td>
<td>650</td>
<td>680</td>
<td>1350</td>
<td>650</td>
</tr>
</tbody>
</table>

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.
36.8 **SAES registers**

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

### 36.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30-28</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPRST</td>
<td>KEYSEL[2:0]</td>
<td>SAES peripheral software reset</td>
</tr>
<tr>
<td></td>
<td>KSHAREID[1:0]</td>
<td>Key selection</td>
</tr>
<tr>
<td></td>
<td>KMOD[1:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NPBLB[3:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>KEYSIZE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Res.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CHMOD[2]</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>Res.</td>
<td>GCMFH[1:0]</td>
<td>DMAOUTEN</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>CHMOD[1:0]</td>
<td>MODE[1:0]</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31 IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be kept low while writing any configuration registers.

**Bits 30:28 KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x3: Application hardware key (AHK)

0x4: XOR of DHUK and BHK

0x5: XOR of DHUK and AHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared. Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.
Bits 27:26 **KSHAREID[1:0]**: Key share identification
This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.
0x0: CRYP peripheral
Others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection
The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.
0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.
0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.
0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.
Others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]**: Number of padding bytes in last block
This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.
0x0: All bytes are valid (no padding)
0x1: Padding for the last LSB byte
... 
0xF: Padding for the 15 LSB bytes of last block.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **KEYSIZE**: Key size selection
This bitfield defines the key length in bits of the key used by SAES.
0: 128-bit
1: 256-bit
When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt. Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]**: GCM or CCM phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase
Bit 12 DMAOUTEN: DMA output enable
This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 DMAINEN: DMA input enable
This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 CHMOD[2:0]: Chaining mode
This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 MODE[1:0]: Operating mode
This bitfield selects the SAES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access.

Bits 2:1 DATATYPE[1:0]: Data type
This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in Section 36.4.16: SAES data registers and data swapping.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.
Bit 0  **EN**: Enable
This bit enables/disables the SAES peripheral.
0: Disable
1: Enable
At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.
This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.
The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:
- KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)
- KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

### 36.8.2 SAES status register (SAES_SR)
Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

Bit 7  **KEYVALID**: Key valid flag
This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.
0: Key not valid
1: Key valid
The EN bit can only be set when KEYVALID is set.
In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.
When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.
If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set.
See the KEIF flag description for more details.
For further information on key loading, refer to Section 36.4.17: SAES key registers.

Bits 6:4 Reserved, must be kept at reset value.
Bit 3 **BUSY**: Busy
This flag indicates whether SAES is idle or busy.
0: Idle
1: Busy
SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed. SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral. When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag
This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.
0: No error
1: Unexpected write to SAES_DINR register occurred during computation or data output phase.
The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.
The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag
This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.
0: No error
1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.
The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.
The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

### 36.8.3 SAES data input register (SAES_DINR)
Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **DIN[31:0]**: Data input
A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.
Reads return zero.
36.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C
Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

DOUT[31:16]: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

36.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

KEY[31:16]: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.
36.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 KEY[63:32]: Cryptographic key, bits [63:32]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

36.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 KEY[95:64]: Cryptographic key, bits [95:64]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

36.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 KEY[127:96]: Cryptographic key, bits [127:96]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.
36.8.9  SAES initialization vector register 0 (SAES_IVR0)
Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>Bits 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **IV[31:0]**: Initialization vector input, bits [31:0]
SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register.

36.8.10  SAES initialization vector register 1 (SAES_IVR1)
Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 63:32</th>
<th>Bits 95:64</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **IV[63:32]**: Initialization vector input, bits [63:32]
Refer to the SAES_IVR0 register for description of the IV[128:0] bitfield.

36.8.11  SAES initialization vector register 2 (SAES_IVR2)
Address offset: 0x028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 95:64</th>
<th>Bits 79:64</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0  **IV[95:64]**: Initialization vector input, bits [95:64]
Refer to the SAES_IVR0 register for description of the IV[128:0] bitfield.
36.8.12  SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  IVI[127:96]: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

36.8.13  SAES key register 4 (SAES_KEYR4)

Address offset: 0x030
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  KEY[159:128]: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

36.8.14  SAES key register 5 (SAES_KEYR5)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  KEY[191:160]: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.
36.8.15 SAES key register 6 (SAES_KEYR6)
Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

36.8.16 SAES key register 7 (SAES_KEYR7)
Address offset: 0x03C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

36.8.17 SAES suspend registers (SAES_SUSPRx)
Address offset: 0x040 + 0x4 * x, (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

1494/3791 RM0477 Rev 6
**36.8.18 SAES interrupt enable register (SAES_IER)**

Address offset: 0x300

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 3 RNGEIE: RNG error interrupt enable**
  - This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.
  - 0: Disabled (masked)
  - 1: Enabled (not masked)

- **Bit 2 KEIE: Key error interrupt enable**
  - This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.
  - 0: Disabled (masked)
  - 1: Enabled (not masked)

- **Bit 1 RWEIE: Read or write error interrupt enable**
  - This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.
  - 0: Disabled (masked)
  - 1: Enabled (not masked)

- **Bit 0 CCFIE: Computation complete flag interrupt enable**
  - This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.
  - 0: Disabled (masked)
  - 1: Enabled (not masked)
36.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304
Reset value: 0x0000 0000

| Bits 31:4 | Reserved, must be kept at reset value. |
| Bit 3    | RNGEIF: RNG error interrupt flag |
|          | This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy). |
|          | 0: RNG bus is functional |
|          | 1: Error detected on RNG bus interface (random seed fetching error) |
|          | The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set. |
|          | The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral. |
| Bit 2    | KEIF: Key error interrupt flag |
|          | This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden. |
|          | 0: No key error detected |
|          | 1: Key information failed to load into key registers or the key register use is forbidden |
|          | The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register. |
|          | The flag is cleared by setting the corresponding bit of the SAES_ICR register. |
|          | KEIF is raised upon any of the following events: |
|          | –SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4). |
|          | –SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4). |
|          | –SAES fails to load the AHK (KEYSEL[2:0] = 0x3 or 0x5). |
|          | –CRYP fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2). |
|          | –SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse). |
|          | KEIF must be cleared by the application software, otherwise KEYVALID cannot be set. |
| Bit 1    | RWEIF: Read or write error interrupt flag |
|          | This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register. |
|          | 0: No read or write error detected |
|          | 1: Read or write error detected |
|          | The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set. |
|          | The flag is cleared by setting the corresponding bit of the SAES_ICR register. |
|          | The flags has no meaning when key derivation mode is selected. |
|          | See the SAES_SR register for details. |
### 36.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RNGEIF: RNG error interrupt flag clear</td>
</tr>
<tr>
<td></td>
<td>Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.</td>
</tr>
<tr>
<td>30</td>
<td>KEIF: Key error interrupt flag clear</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the KEIF status bit of the SAES_ISR register.</td>
</tr>
<tr>
<td>29</td>
<td>RWEIF: Read or write error interrupt flag clear</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_ISR register.</td>
</tr>
<tr>
<td>28</td>
<td>CCF: Computation complete flag clear</td>
</tr>
<tr>
<td></td>
<td>Setting this bit clears the CCF status bit of the SAES_ISR register.</td>
</tr>
</tbody>
</table>

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

### 36.8.21 SAES register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 336. SAES register map and reset values
### Table 336. SAES register map and reset values (continued)

| Offset | Register name | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | BUSY | WRERRF | RDERRF |
|--------|---------------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|-----|-------|--------|
| 0x004  | SAES_SR       |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | SAES_DINR     | DIN[31:0]     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x00C  | SAES_DOUTR    | DOUT[31:0]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x010  | SAES_KEYR0    | KEY[31:0]     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x014  | SAES_KEYR1    | KEY[63:32]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x018  | SAES_KEYR2    | KEY[95:64]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x01C  | SAES_KEYR3    | KEY[127:96]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x020  | SAES_IVR0     | IV[31:0]      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x024  | SAES_IVR1     | IV[63:32]     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x028  | SAES_IVR2     | IV[95:64]     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x02C  | SAES_IVR3     | IV[127:96]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x030  | SAES_KEYR4    | KEY[159:128]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x034  | SAES_KEYR5    | KEY[191:160]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x038  | SAES_KEYR6    | KEY[223:192]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x03C  | SAES_KEYR7    | KEY[255:224]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x040  | SAES_SUSPR0   | SUSP[128:96]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x044  | SAES_SUSPR1   | SUSP[31:0]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x048  | SAES_SUSPR2   | SUSP[31:0]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Refer to Section 2.3 on page 149 for the register boundary addresses.
37 Cryptographic processor (CRYP)

37.1 Introduction

The cryptographic processor (CRYP) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. CRYP supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128, 192 or 256 bits. CRYP has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports both single and fixed DMA burst transfers for incoming and outgoing data (two DMA channels are required). CRYP also includes input and output FIFOs for better performance.

37.2 CRYP main features

- Compliant with NIST FIPS publication 197 “Advanced encryption standard (AES)” (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 16-byte data block processing, supporting cipher key lengths of 128, 192 and 256 bits
  - 14 or 18 clock cycle latency in ECB mode for processing one 16-byte block with 128-bit or 256-bit key, respectively
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing the cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit input buffer associated with an internal input FIFO of eight 32-bit words, corresponding to two AES blocks
- 32-bit output buffer associated with an internal output FIFO of eight 32-bit words, corresponding to two AES blocks
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. The output FIFO supports both single and burst transfers, while the input FIFO supports only burst transfers.
- Data swapping logic to support 1-, 8-, 16- or 32-bit data
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only. Other access types generates an AHB error, and write accesses are ignored.
- Possibility for software to suspend a message if CRYP needs to process another message with a higher priority, then resume the original message
37.3 CRYP implementation

The devices have one CRYP peripheral, implemented as per the following table. It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 337. CRYP versus SAES features

<table>
<thead>
<tr>
<th>Modes or features(1)</th>
<th>CRYP</th>
<th>SAES</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECB, CBC chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>CTR, CCM, GCM chaining</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AES 128-bit ECB encryption in cycles</td>
<td>14</td>
<td>480</td>
</tr>
<tr>
<td>DHUK and BHK key selection</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Resistance to side-channel attacks</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Shared key between SAES and CRYP</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Key size in bits</td>
<td>128, 192, 256</td>
<td>128, 256</td>
</tr>
</tbody>
</table>

1. X = supported.
37.4 CRYP functional description

37.4.1 CRYP block diagram

The figure below shows the block diagram of the cryptographic processor.

Figure 37.2. CRYP block diagram
37.4.2 CRYP internal signals

Table 338 describes the user relevant internal signals interfacing the CRYP peripheral.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cryp_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>cryp_it</td>
<td>Digital output</td>
<td>CRYP interrupt request</td>
</tr>
<tr>
<td>cryp_in_dma</td>
<td>Digital input/output</td>
<td>CRYP incoming data FIFO DMA request/acknowledge</td>
</tr>
<tr>
<td>cryp_out_dma</td>
<td>Digital input/output</td>
<td>CRYP processed data FIFO DMA request/acknowledge</td>
</tr>
</tbody>
</table>

37.4.3 CRYP reset and clocks

The CRYP peripheral is clocked by the AHB clock.
The CRYP has a dedicated reset bit in the RCC.

37.4.4 CRYP symmetric cipher implementation

The cryptographic processor (CRYP) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in Table 339. Those functions can be certified NIST PUB 140-3.

<table>
<thead>
<tr>
<th>Operations</th>
<th>Algorithm</th>
<th>Specification</th>
<th>Key size in bits</th>
<th>Chaining modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption, decryption</td>
<td>AES</td>
<td>FIPS PUB 197</td>
<td>128, 192, 256</td>
<td>ECB, CBC, CTR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NIST SP800-38A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Authenticated encryption or decryption</td>
<td>AES</td>
<td>NIST SP800-38C</td>
<td></td>
<td>GCM, CCM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NIST SP800-38D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cipher-based message authentication code</td>
<td>AES</td>
<td>NIST SP800-38D</td>
<td></td>
<td>GMAC</td>
</tr>
</tbody>
</table>

CRYP can be used directly by the CPU, or indirectly using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any CRYP processing, following the sequence described in Section 37.4.8.
37.4.5 **CRYP encryption/ decryption typical usage**

The following figure shows a typical operation for encryption or decryption.

![Figure 373. Encryption/ decryption typical usage](MSv66120V1)

**Initialization**

The CRYP peripheral is initialized according to the chaining mode. Refer to **Section 37.4.9: CRYP basic chaining modes (ECB, CBC)** and **Section 37.4.10: CRYP counter mode (CTR)** for details.

**Data append**

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to **Section 37.4.7: CRYP ciphertext stealing and data padding** if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

**General conditions for data processing**

The CRYP is busy and processing data (BUSY flag set) when all the conditions below are met:

- CRYPEN is set in CRYP_CR register.
- There are enough data in the input FIFO (at least four words).
- There is enough free-space in the output FIFO (at least four word locations).

It is possible to clear the CRYPEN bit while BUSY bit is set. In this case the ongoing AES processing first completes (that is, the word results are written to the output FIFO) before the BUSY bit is cleared by hardware.

*Note:* If the application needs to suspend a message to process another one with a higher priority, refer to **Section 37.4.8: CRYP suspend and resume operations**.
Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. When KEYVALID is set, enable CRYP by setting the CRYPEN bit of the CRYP_CR register (if not already done).

2. Repeat the following subsequence until the payload is entirely processed:
   a) Write data in the input FIFO (one block or until the FIFO is full).
   b) Repeat until the second last block of data has been processed:
      – Wait until the not-empty flag OFNE is set, then read the output FIFO (one block or until the FIFO is empty).
      – Wait until the not-full flag IFNF is set, then write the input FIFO (one block or until the FIFO is full), except if it is the last block.
   c) The BUSY bit is set automatically by CRYP. At the end of the processing, the BUSY bit is cleared and both FIFOs are empty (input FIFO empty flag IFEM is set, output FIFO not-empty flag OFNE is cleared).
   d) If the next processing block is the last block, pad the data with zeros to obtain a complete block (when applicable), and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). This operation must be performed after checking that the BUSY bit is cleared.

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the CRYP peripheral by clearing CRYPEN.

Note: NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in Interrupt mode

The method uses interrupt from the CRYP peripheral to control the data append, through the following sequence:

1. Enable interrupts from CRYP, by setting the INIM and OUTIM bits of the CRYP_IMSCR register.

2. When KEYVALID is set, enable the CRYP peripheral, by setting CRYPEN (if not already done).

3. Handle the data in the CRYP interrupt service routine that manages the input FIFO. Upon each interrupt:
   a) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0] bitfield) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). This operation must be performed after checking that the BUSY bit is cleared. After that load the block into the input FIFO.
   b) If the next processing block is not the last block, load the data into the input FIFO. Application can load only one block (4 words), or load data until the input FIFO is full.
   c) After the last word of data has been written, disable the input FIFO interrupts by clearing the INIM bit (if applicable), then quit the interrupt service routine.
4. Handle the data in the CRYP interrupt service routine that manages the output FIFO. Upon each interrupt:
   a) Read data from the output FIFO. Application can read only one block (4 words), or read data until the FIFO is empty.
   b) When the last word of the message has been read, INIM and BUSY bits are cleared and both FIFOs are empty (IFEM is set, OFNE is cleared). Disable the output FIFO interrupt by clearing the OUTIM bit, and disable the peripheral by clearing the CRYPEN bit.
   c) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then quit the interrupt service routine.

Note: The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC and CCM chaining modes.

Appending data using the DMA

With this method, all the transfers and processing are managed by DMA and CRYP. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the CRYP peripheral input and the processed data from the CRYP peripheral output to the memory, as described in Section 37.6: CRYP DMA requests. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer must not include the last block. The sequence described in Appending data using the CPU in polling mode must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for CRYP to compute a correct tag.
3. When KEYVALID is set, enable the CRYP peripheral, by setting CRYPEN (if not already done).
4. Enable DMA requests, by setting DIEN and DOEN of the CRYP_DMACR register.
5. Upon DMA interrupt indicating the transfer completion, get the CRYP-processed data from the memory. Both FIFOs are normally empty and BUSY flag is cleared.

Note: The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC and CCM chaining modes.

Caution: It is important that the DMA controller empties the CRYP output FIFO before filling up the CRYP input FIFO. To achieve this, the DMA controller must be configured so that the transfer from the CRYP to the memory has a higher priority than the transfer from the memory to the CRYP.
37.4.6 CRYP authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

![Figure 374. Typical operation with authentication](image)

Section 37.4.11: CRYP AES Galois/counter mode (GCM) and Section 37.4.13: CRYP AES Counter with CBC-MAC (CCM) describe detailed sequences supported by CRYP.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by CRYP is described in Section 37.4.12: CRYP AES Galois message authentication code (GMAC).

37.4.7 CRYP ciphertext stealing and data padding

When using ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode. Since CRYP does not implement such techniques, the application must complete the last block of input data using data from the second last block.

**Note:** Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in other modes than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since CRYP does not implement automatic data padding operation to the last block, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.
37.4.8 CRYP suspend and resume operations

A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as CRYP is enabled again, to receive a next data block. Figure 375 gives an example of suspend and resume operation: message 1 is suspended in order to send a shorter and higher-priority message 2.

![Figure 375. Example of suspend mode management](image)

Detailed descriptions of suspend and resume operations are provided in the sections dedicated to each chaining mode.

37.4.9 CRYP basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB mode, a special key scheduling is required before processing the first block.

Figure 376 and Figure 377 describe the electronic codebook (ECB) chaining implementation in encryption and decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x4.
In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC mode, a special key scheduling is required before processing the first block.
Figure 378 and Figure 379 describe the cipher block chaining (CBC) implementation in encryption and decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x5.

For more details, refer to NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.
ECB/CBC encryption process

This process is described in Section 37.4.5, with the following sequence of events:

1. Disable the CRYP peripheral by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x4 or 0x5) in encryption mode (ALGODIR = 0).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode using KMOD[1:0]: if the key is coming from SAES peripheral, set KMOD[1:0] to 0x2, else keep it at 0x0.
4. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
5. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0] = 0x2, the key is being transferred from the SAES peripheral (see Section 37.4.14).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable CRYP by setting CRYPEN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in Section 37.4.7.
   b) Append the cleartext block into CRYP as described in Section 37.4.5, then read the CRYP_DOUTR register to save the ciphertext block(s), MSB first.
   c) Repeat the step b) until the third-last plaintext block is encrypted. For the last two blocks, follow the steps a) and b).
9. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above. KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time.
ECB/CBC decryption process

This process is described in Section 37.4.5, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select the key derivation mode (write ALGOMODE[3:0] with 0x7) in encryption mode (ALGODIR = 0).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2, else keep it at 0x0.
4. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0] = 0x2, the key is being transferred from the SAES peripheral (see Section 37.4.14).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the CRYP peripheral by setting CRYPEN. The peripheral immediately starts an AES round for key preparation, setting the BUSY bit.
7. Wait until the BUSY bit is cleared, then select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x4 or 0x5) in decryption mode (ALGODIR = 1). Do not change other parameters.
8. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
9. Enable CRYP by setting CRYPEN.
10. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in Section 37.4.7.
    b) Append the ciphertext block into CRYP as described in Section 37.4.5, then read the CRYP_DOUTR register to save the cleartext block(s), MSB first.
    c) Repeat the step b) until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps a) and b).
11. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.
KM0D[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time.

Suspend and resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO by clearing the DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. If DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP peripheral, by clearing CRYPEN.
5. Save the CRYP_CR register and clear the key registers if they are not needed, to process the higher-priority message.
6. Save initialization vector registers (only required in CBC mode as the CRYP_IVxL/R registers are altered during the data processing).
7. If DMA is used, save the DMA controller status (pointers for CRYP input and output data transfers, number of remaining bytes, and so on).

**To resume the processing of a message,** proceed as follows:
1. If DMA is used, configure the DMA controller so as to complete the remaining input FIFO and output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the CRYP_CR register (with correct KEYSIZE), then restore the CRYP_KxL/R registers. If KMOD[1:0] = 0x2 the key needs to be transferred again from the SAES peripheral (see Section 37.4.14).
4. Prepare the decryption key as described in **ECB/CBC decryption process** (only required for ECB or CBC decryption).
5. Restore the CRYP_IVxL/R registers, using the saved configuration (only required in CBC mode).
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA transfers, by setting DIEN and DOEN.

**Note:** Key registers do not need to be saved as the original key value is known by the application.

### 37.4.10 CRYP counter mode (CTR)

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in **Figure 380**.

**Figure 380. Message construction in CTR mode**
The structure of this message is:

- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IVI): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is complete. The initial value of the counter must be set.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.

**CTR encryption and decryption**

*Figure 381* describes the counter (CTR) chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x6.

**Figure 381. CTR encryption**

Initialization vectors in CRYP must be initialized as shown in *Table 340*.

**Table 340. Counter mode initialization vector definition**

<table>
<thead>
<tr>
<th>CRYP_IV0LR[31:0]</th>
<th>CRYP_IV0RR[31:0]</th>
<th>CRYP_IV1LR[31:0]</th>
<th>CRYP_IV1RR[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>32-bit counter = 0x1</td>
</tr>
</tbody>
</table>
CTR encryption and decryption process

This process is described in Section 37.4.5, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select CTR chaining mode (write ALGOMODE[3:0] with 0x6) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
4. Write the initialization vector into the CRYP_IVxL/R registers according to Table 340.
5. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2 the key is being transferred from the SAES peripheral (see Section 37.4.14).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable CRYP by setting CRYPEN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into CRYP as described in Section 37.4.5, then read the CRYP_DOUTR register to save the resulting block(s), MSB first.
   c) Repeat the step b) until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time.

Suspend and resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in Section 37.4.9: CRYP basic chaining modes (ECB, CBC).

Note: Like for CBC mode, IVI registers must be reloaded during the resume operation.

37.4.11 CRYP AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) enables encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.
The structure of this message is defined as below:

- **16-byte Initial Counter Block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV)**: a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is complete. According to NIST specification, the counter value is 0x2 when processing the first block of payload.

- **Authenticated header AAD** (also known as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} - 1$ bits. This part of the message is only authenticated, not encrypted.

- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-bytes blocks.

- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in Table 342.

The GCM standard specifies that ciphertext C has the same bit length as the plaintext P. A special padding scheme is required when a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes.

For more details, refer to NIST Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.

Figure 383 describes the GCM chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x6.
Figure 383. GCM authenticated encryption

The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in Table 341.

Table 341. GCM mode IVI registers initialization

<table>
<thead>
<tr>
<th>Register</th>
<th>CRYP_IV0LR[31:0]</th>
<th>CRYP_IV0RR[31:0]</th>
<th>CRYP_IV1LR[31:0]</th>
<th>CRYP_IV1RR[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>32-bit counter = 0x2</td>
</tr>
</tbody>
</table>

The last block of a GCM message contains the AAD header length and the payload length information, as shown in Table 342.

Table 342. GCM last block definition

<table>
<thead>
<tr>
<th>Word order to CRYP_DINR</th>
<th>First word</th>
<th>Second word</th>
<th>Third word</th>
<th>Fourth word</th>
</tr>
</thead>
</table>
GCM encryption and decryption process

This process is described in Section 37.4.6, with the following sequence of events:

**GCM initialize**
1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register:
   a) Select GCM chaining mode (write ALGOMODE[3:0] with 0x8) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0]
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
   e) Select the GCM initialization phase, by writing 0x0 to GCM_CCMPH[1:0]
4. Write the initialization vector in CRYP_IVxL/R registers according to Table 341.
5. Write the key in CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2, the key is being transferred from the SAES peripheral (see Section 37.4.14).
6. Wait until KEYVALID is set (the key loading completed).
7. Set CRYPEN to start the calculation of the hash key. CRYPEN is automatically cleared when the calculation is complete.
8. Wait for the CRYPEN bit to be cleared, indicating that the GCM hash subkey (H) computation is complete.

**GCM header phase**
9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCM_CCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Set CRYPEN to start accepting data.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in Section 37.4.5.
    c) Repeat the step b) until the second-last AAD data block is processed. For the last block follow the steps a) and b).
11. Wait until the BUSY flag is cleared.

*Note:* This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.
GCM payload phase

12. Initialize payload phase:
   a) Disable the CRYP peripheral, by clearing CRYPEN.
   b) Select the GCM payload phase, by writing GCM_CCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
   c) Enable the CRYP peripheral by setting CRYPEN, to start accepting data.

13. Append payload data:
   a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data into CRYP as described in Section 37.4.5, then read the CRYP_DOUTR register to save the resulting block(s).
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

14. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload, that is, Len(C) = 0 (see GMAC mode).

GCM finalization

15. Select the GCM final phase, by writing GCM_CCMPH[1:0] with 0x3, and clear the ALGODIR bit (encryption). Do not change the other configurations written during GCM initialization.

16. Write the final GCM block into the CRYP_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in Table 342.

17. Wait until the OFNE flag in the CRYP_SR register is set.

18. Get the GCM authentication tag, by reading the CRYP_DOUTR register four times.

19. Disable the CRYP peripheral, by clearing CRYPEN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to CRYP_DINR normally (no swapping).
Suspend and resume operations in GCM mode

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO, by clearing DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. In the payload phase, if DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP by clearing CRYPEN.
5. Save the current CRYP_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
6. In the payload phase, save the CRYP_IVxL/R registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
7. Save the CRYP_CSGCMCCMRx and CRYP_CSGCMRx registers in memory.
8. If DMA is used, save the DMA controller status (pointer for CRYP input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for CRYP output data transfers.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the initial setting values in the CRYP_CR and CRYP_KxL/R registers. If KMOD[1:0]= 0x2 the key needs to be transferred again from the SAES peripheral (see Section 37.4.14).
4. Write the suspend register values, previously saved in the memory, back into their corresponding CRYP_CSGCMCCMRx and CRYP_CSGCMRx registers.
5. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding CRYP_IVxL/R registers. In the header phase, write initial setting values back into the CRYP_IVxL/R registers.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA requests, by setting DIEN (and DOEN if in payload phase).
37.4.12 CRYP AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) enables the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

**Figure 384. Message construction in GMAC mode**

For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

**Figure 385** describes the GMAC chaining implementation in the CRYP peripheral. To select this chaining mode, write ALGOMODE[3:0] with 0x6.

**Figure 385. GMAC authentication mode**
The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as for the GCM, except that the payload phase is omitted.

**Suspend and resume operations in GMAC**

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

### 37.4.13 CRYP AES Counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm enables encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

**Note:** *NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.*

The following figure shows typical message construction for CCM.

**Figure 386. Message construction in CCM mode**

The structure of the message is:

- **16-byte first authentication block (B0),** composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (**Len(P)**)
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of **Len(N)** size. The sum **Len(N) + Len(P)** must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that **Len(P)** < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-bytes blocks (B)** associated to the Associated Data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see Figure 386). The standard also states that, on the MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as $0xFF \ || \ 0xFE \ || \ [a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as $0xFF \ || \ 0xFF \ || \ [a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see Figure 386).

- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

**Note:** *CCM chaining mode can also be used with associated data only (that is, no payload).*

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

- **N:** 10111213 141516 (Len(N) = 56 bits or 0x7 bytes)
- **A:** 00010203 04050607 (Len(A) = 64 bits or 0x8 bytes)
- **P:** 20212223 (Len(P) = 32 bits i.e. Q = 0x4 bytes)
- **T:** 6084341b (Len(T) = 32 bits or t = 4)
- **B0:** 4f101112 13141516 00000000 00000004
- **B1:** 00080001 02030405 06070000 00000000
- **B2:** 20212223 00000000 00000000 00000000
- **CTR0:** 0710111213 141516 00000000 00000000
- **CTR1:** 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality.
Figure 387 describes the CCM chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x9.

The first block of a CCM message (B0) must be prepared by the application as defined in Table 343.

Table 343. CCM mode IVI registers initialization

<table>
<thead>
<tr>
<th>Register</th>
<th>CRYP_IV0LR[31:0]</th>
<th>CRYP_IV0RR[31:0]</th>
<th>CRYP_IV1LR[31:0]</th>
<th>CRYP_IV1RR[31:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>B0[127:96]^{(1)}</td>
<td>B0[95:64]</td>
<td>B0[63:32]</td>
<td>B0[31:0]^{(2)}</td>
</tr>
</tbody>
</table>

1. The 5 most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for bit 0 that is set.

CRYP supports counters up to 64 bits, as specified by NIST.
CCM encryption and decryption process

This process is described in Section 37.4.6, with the following sequence of events:

CCM initialize
1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register:
   a) Select CCM chaining mode (write ALGOMODE[3:0] with 0x9) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0]
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
   e) Select the CCM initialization phase, by writing 0x0 to GCM_CCMPH[1:0]
4. Write the initialization vector into the CRYP_IVxL/R registers with CTR1 information, as defined in Table 343
5. Write the key in CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2 the key is being transferred from the SAES peripheral (see Section 37.4.14).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the CRYP peripheral by setting CRYPEN.
8. Write the B0 data in CRYP_DINR registers according to Table 343, then wait for the CRYPEN bit to be cleared.

Note: In this initialization phase, data have to be inserted normally (no swapping).

CCM header phase
9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the GCM header phase, by writing GCM_CCMPH[1:0] with 0x1. Do not change the other configurations written during the GCM initialization.
   c) Set CRYPEN to start accepting data.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in Section 37.4.5.
    c) Repeat the step b) until the second-last associated data block is processed. For the last block follow the steps a) and b).
11. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.
CCM payload phase

12. Initialize payload phase:
   a) Disable the CRYP peripheral, by clearing CRYPEN.
   b) Select the GCM payload phase, by writing GCM_CCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
   c) Enable the CRYP peripheral by setting CRYPEN, to start accepting data.

13. Append payload data:
   a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data into CRYP as described in Section 37.4.5, then read the CRYP_DOUTR register to save the resulting block(s).
   c) Repeat the step b) until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps a) and b). For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

14. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).
Remove LSB_{Len(T)} encrypted tag information when decrypting ciphertext C.

CCM finalization

15. Select the GCM final phase, by writing GCM_CCMPH[1:0] with 0x3, and clear the ALGODIR bit (encryption). Do not change the other configurations written during GCM initialization.

16. Write the CTR0 information into the CRYP_DINR register. CTR0 information is described in Table 343, with the bit[0] cleared.

17. Wait until the OFNE flag in the CRYP_SR register is set.

18. Get the CCM authentication tag, by reading the CRYP_DOUTR register four times.

19. Disable the CRYP peripheral by clearing CRYPEN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In this final phase, data have to be inserted normally (no swapping).
Suspend and resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO, by clearing DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. In the payload phase, if DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP by clearing CRYPEN.
5. Save the current CRYP_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
6. In the payload phase, save the CRYP_IVxL/R registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
7. Save the CRYP_CSGCMCCMRx registers in the memory.
8. If DMA is used, save the DMA controller status (pointer for CRYP input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for CRYP output data transfers.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the initial setting values in the CRYP_CR and CRYP_KxL/R registers. If KMOD[1:0]= 0x2 the key needs to be transferred again from the SAES peripheral (see Section 37.4.14).
4. Write the suspend register values, previously saved in the memory, back into their corresponding CRYP_CSGCMCCMRx registers.
5. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding CRYP_IVxL/R registers. In the header phase, write initial setting values back into the CRYP_IVxL/R registers.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA requests, by setting DIEN (and DOEN if in payload phase).

37.4.14 AES key sharing with secure AES co-processor

The CRYP peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral. When it is ready, the CRYP application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in section SAES operations with shared keys in section Secure AES coprocessor (SAES) of this document.
1. In SAES peripheral, the application encrypts the key to share in shared-key mode (KMOD[1:0] at 0x2). When applicable, the proper CRYP instance must be selected through KSHAREID[1:0].

2. Each time the shared key is required in the CRYP peripheral, the application decrypts it in the SAES peripheral in shared-key mode (KMOD[1:0] at 0x2). When applicable, the proper CRYP instance must be selected through KSHAREID[1:0].

3. Once the shared key is decrypted and loaded in SAES_KEYRx registers it can be shared with CRYP. To load the shared key in CRYP, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transferred by hardware into the CRYP_KxL/R registers and the BUSY flag in the CRYP_SR register set.

4. Once the key transfer is complete, the BUSY flag is cleared and the KEYVALID flag set in the CRYP_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KERF flag is set in the CRYP_SR register, it means that the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID in SAES_SR cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is complete, the CRYP is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in CRYP peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

When sharing keys KEYSIZE in CRYP must be identical to SAES value (128 or 256 bits).

37.4.15 CRYP data registers and data swapping

Data input and output

A 16-byte data block enters the CRYP peripheral with four successive 32-bit word writes into the CRYP_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the CRYP peripheral with four successive 32-bit word reads of the CRYP_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See Table 344 “no swapping” option for details.

Data swapping

The CRYP peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the CRYP input FIFO, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the CRYP output FIFO. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0]. The selection applies both to the input and the output of the CRYP, except in GCM/CCM final phase (data must be inserted normally).
Note: The data in CRYP key registers (CRYP_KxL/R) and initialization vector registers (CRYP_IVxL/R) are not sensitive to the swap mode selection.

The CRYP data swapping feature is summarized in Table 344 and Figure 388.

<table>
<thead>
<tr>
<th>DATATYPE[1:0]</th>
<th>Swapping performed</th>
<th>Data block</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>System memory data (big-endian)</td>
</tr>
<tr>
<td>0x0</td>
<td>No swapping</td>
<td>Block[127..64]: 0x04E6F772 2E04CE96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[63..0]: 0x4E6F7720 69732074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[127..96]: 0x04EE F672</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[95..64]: 0x2E04 CE96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x8, word[63..32]: 0x4E6F 7720</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0xC, word[31..0]: 0x6973 2074</td>
</tr>
<tr>
<td>0x1</td>
<td>Half-word (16-bit) swapping</td>
<td>Block[63..0]: 0x4E6F 7720 6973 2074</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x7720 4E6F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2074 6973</td>
</tr>
<tr>
<td>0x2</td>
<td>Byte (8-bit) swapping</td>
<td>Block[63..0]: 0x4E 6F 77 20 69 73 20 74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x2077 6F4E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x7420 7369</td>
</tr>
<tr>
<td>0x3</td>
<td>Bit swapping</td>
<td>Block[63..32]: 0x4E6F 7720 0100 1110 0110 1111 0111 0010 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Block[31..0]: 0x6973 2074 0110 1001 0111 0011 0010 0000 0111 0100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110</td>
</tr>
</tbody>
</table>
Data padding

Figure 388 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3
37.4.16 CRYP key registers

The eight CRYP_KxR/LR write-only registers store the encryption or decryption key information, as shown in Table 345. Reads are not allowed for security reason.

Note: In memory and in CRYP key registers, keys are stored in big-endian format, with most significant byte on the lowest address.

| Table 345. Key endianness in CRYP_KxR/LR registers (128/192/256-bit keys) |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| -               | -               | -               | -               | k[127:96]      | k[95:64]       | k[63:32]       | k[31:0]        |

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the CRYP_KxL/R registers are ignored when BUSY bit is set. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes (in ascending order):

- K2LR, K2RR, K3LR, then K3RR registers for KEYSIZE = 0x0
- K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for KEYSIZE = 0x1
- K0LR, K0RR, K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for KEYSIZE = 0x2

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID bit is cleared. Once the key registers writing sequence is complete, KEYVALID is set and CRYPEN bit becomes writable. If an error occurs, KEYVALID is cleared and KERF set (see Section 37.4.18).

37.4.17 CRYP initialization vector registers

The four CRYP_IVxR/LR registers store the initialization vector (IVI) information, as shown in Table 346. They can only be written if BUSY bit is cleared.

Note: In memory and in CRYP IVI registers, initialization vectors are stored in big-endian format, with most significant byte on the lowest address.

| Table 346. Initialization vector endianness in CRYP_IVxR registers (AES) |
|-----------------|-----------------|-----------------|-----------------|
| CRYP_IV0L[31:0] | CRYP_IV0R[31:0] | CRYP_IV1L[31:0] | CRYP_IV1R[31:0] |

Initialization vector information depend on the chaining mode selected. When used, CRYP_IVxR/LR registers are updated upon each AES computation cycle (useful for
managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled by DATATYPE[1:0].

### 37.4.18 CRYP error management

The CRYP peripheral manages the errors described in this section.

**Key error flag (KERF)**

There are multiple sources of errors that set the KERF flag and clear the KEYVALID bit of the CRYP_SR register:

- **Key writing sequence error**: triggered upon detecting an incorrect sequence of writing key registers. See Section 37.4.16: CRYP key registers for details.
- **Key sharing size mismatch error**: triggered when KMOD[1:0] is at 0x2 and KEYSIZE in CRYP does not match KEYSIZE in SAES peripheral, or KEYSIZE is at 0x1 in CRYP_CR (192-bit key).
- **Key sharing error**: triggered upon failing transfer of SAES shared key to CRYP. See Section 37.4.14: AES key sharing with secure AES co-processor for details.

Upon a key writing sequence error, KERF flag is cleared when a correct key writing sequence starts.

Upon a key sharing error, reset both CRYP and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

>Note: For any key error, clear KERF flag prior to setting CRYPEN.

### 37.5 CRYP interrupts

There are two individual maskable interrupt sources generated by the CRYP peripheral to signal the following events:

- Input FIFO empty (IFEM) or not full (IFNF)
- Output FIFO full (OFFU) or not empty (OFNE)

These sources are combined into a common interrupt signal from the CRYP peripheral that connects to the Arm® Cortex® interrupt controller. The application can enable or disable CRYP interrupt sources individually by setting/clearing the corresponding mask bit of the CRYP_IMSCR register. The status of the individual source of event flags can be read from the CRYP_SR register.

The global status of the two source of interrupts can be read either from the CRYP_RISR register (raw, that is unmasked), or from the CRYP_MISR register (masked).

*Table 347* gives a summary of the available features.
Output FIFO service interrupt (OUTMIS)

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is, when the interrupt follows the state of the output FIFO not empty flag OFNE).

The output FIFO service interrupt OUTMIS is active also when CRYP is disabled. Consequently, when CRYPEN bit is cleared, the OUTMIS bit is not cleared when the output FIFO is not empty.

Input FIFO service interrupt (INMIS)

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is active only when CRYP is enabled. Consequently, when CRYPEN bit is cleared, the INMIS bit stay low even if the input FIFO is empty.

37.6 CRYP DMA requests

The CRYP peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DIEN and DOEN bits of the CRYP_DMACR register.

Detailed usage of DMA with CRYP can be found in Appending data using the DMA subsection of Section 37.4.5: CRYP encryption/decryption typical usage.
Data input using DMA

Setting DIEN enables DMA writing into CRYP. The CRYP then initiates, during the input phase, a DMA request for each 16-byte data block to write to the CRYP_DINR register (quadruple 32-bit word, MSB first).

The CRYP supports both single and burst DMA transfers. Recommended configuration is fixed burst size of four words.

CRYP issues a DMA transfer request to its input FIFO every time a 16-bytes block has been read from the FIFO.

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 37.4.7).

Data output using DMA

Setting DOEN enables DMA reading from CRYP. The CRYP then initiates, during the output phase, a DMA request for each 16-byte data block to read from the CRYP_DOUTR register (quadruple 32-bit word, MSB first).

The CRYP supports both single and burst DMA transfers. Recommended configuration is fixed burst size of four words.

CRYP issues a DMA transfer request from output FIFO every time one 16-byte block has been written into the FIFO.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when CRYP is disabled (CRYPEN cleared) or the DMA enable bit (DIEN for input data, DOEN for output data) is cleared.

37.7 CRYP processing time

The following tables provide the 16-byte data block processing latency per operating mode.

<table>
<thead>
<tr>
<th>Key size</th>
<th>Mode of operation</th>
<th>Chaining algorithm</th>
<th>Clock cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 bits</td>
<td>encryption or decryption(^{(1)})</td>
<td>ECD, CBC, CTR</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>key preparation</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>192 bits</td>
<td>encryption or decryption(^{(1)})</td>
<td>ECD, CBC, CTR</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>key preparation</td>
<td>-</td>
<td>14</td>
</tr>
<tr>
<td>256 bits</td>
<td>encryption or decryption(^{(1)})</td>
<td>ECD, CBC, CTR</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>key preparation</td>
<td>-</td>
<td>16</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Excluding key preparation time (ECB and CBC only).
37.8 **CRYP registers**

These registers are accessible through 32-bit word single accesses only (otherwise an AHB error is generated, and write accesses are ignored).

### 37.8.1 CRYP control register (CRYP_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>IPRST: CRYP peripheral software reset</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>Setting the bit resets the CRYP peripheral, putting all registers to their default values, except the IPRST bit itself. This bit must be kept cleared while writing any configuration registers.</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>KMOD[1:0]: Key mode selection</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>KEYSIZE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>DATATYPE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>ALGOMODE[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>GCM_CCMPH[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>ALGODIR</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>KEYSIZE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>DATATYPE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>ALGOMODE[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>FFLUSH</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>CRYPEN</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>KEYSIZE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>DATATYPE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>ALGOMODE[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>GCM_CCMPH[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>ALGODIR</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>KEYSIZE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>DATATYPE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>ALGOMODE[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>GCM_CCMPH[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>ALGODIR</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>KEYSIZE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>DATATYPE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>ALGOMODE[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>GCM_CCMPH[1:0]</td>
<td>rw</td>
</tr>
</tbody>
</table>

Attempts to write the bitfield are ignored when BUSY is set.
Bits 23:20 **NPBLB[3:0]**: Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)
0x1: Padding for the last LSB byte
...
0xF: Padding for the 15 LSB bytes of last block.

Attempts to write the bitfield are ignored when BUSY is set.

Bit 18 **Reserved, must be kept at reset value.**

Bits 17:16 **GCM_CCMPH[1:0]**: GCM or CCM phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.

0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Attempts to write the bitfield are ignored when BUSY is set.

Bit 15 **CRYPEN**: CRYP enable

This bit enables/disables the CRYP peripheral.

0: CRYP disabled
1: CRYP enabled

This bit is automatically cleared by hardware upon the completion of the key preparation (ALGOMODE[3:0] at 0x7) and upon the completion of GCM/GMAC/CCM initialization phase. The bit cannot be set as long as KEYVALID is cleared.

Bit 14 **FFLUSH**: FIFO flush

This bit enables/disables the flushing of CRYP input and output FIFOs. Reading this bit always returns 0.

0: No effect
1: FIFO flush enabled

When CRYPEN is cleared, writing this bit to 1 flushes both input and output FIFOs (that is read and write pointers of the FIFOs are reset).

FFLUSH bit must be set when BUSY is cleared, otherwise the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a non-empty FIFO condition.

Attempts to write FFLUSH are ignored when CRYPEN is set.

Bits 13:10 **Reserved, must be kept at reset value.**

Bits 9:8 **KEYSIZE[1:0]**: Key size selection

This bitfield defines the key length in bits of the key used by CRYP.

0x0: 128-bits
0x1: 192 bits
0x2: 256 bits
0x3: Reserved

When KEYSIZE is changed, KEYVALID bit is cleared.

Attempts to write the bitfield are ignored when BUSY is set.
Bits 7:6  **DATATYPE[1:0]: Data type**  
This bitfield defines the format of data written in the CRYP_DINR register or read from the CRYP_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 37.4.15: CRYP data registers and data swapping.*  
0x0: No swapping (32-bit data).  
0x1: Half-word swapping (16-bit data).  
0x2: Byte swapping (8-bit data).  
0x3: Bit-level swapping.  
Attempts to write the bitfield are ignored when BUSY is set.  

Bits 19, 5:3  **ALGOMODE[3:0]: Algorithm mode**  
This bitfield selects the AES algorithm/chaining mode.  
0x4: Electronic codebook (ECB)  
0x5: Cipher Block Chaining (CBC)  
0x6: Counter mode (CTR)  
0x7: AES key preparation for ECB or CBC decryption  
0x8: Galois counter mode (GCM) and Galois message authentication code (GMAC)  
0x9: Counter with CBC-MAC (CCM)  
Others: Reserved  
Attempts to write the bitfield are ignored when BUSY is set.  

Bit 2  **ALGODIR:** Algorithm direction  
This bit selects the algorithm direction.  
0: Encryption  
1: Decryption  
Attempts to write the bitfield are ignored when BUSY is set.  

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

### 37.8.2  **CRYP status register (CRYP_SR)**  
Address offset: 0x04  
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:8  Reserved, must be kept at reset value.

Bit 7  KEYVALID: Key valid flag
   This read-only bit is set by hardware when the key of size defined by KEYSIZE is loaded in
   CRYP_KxR/LR key registers.
   0: Key not valid
   1: Key valid
   The CRYPEN bit can only be set when KEYVALID is set.
   In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the
   correct sequence, otherwise the KERF flag is set and KEYVALID remains cleared.
   When KMOD[1:0] is different from zero, the BUSY flag is automatically set by CRYP. When
   the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KERF is
   set, BUSY cleared and KEYVALID remains cleared.
   If set, KERF must be cleared, otherwise KEYVALID cannot be set. For further information on
   key loading, refer to Section 37.4.16: CRYP key registers.

Bit 6  KERF: Key error flag
   This read-only bit is set by hardware when key information failed to load into key registers.
   0: No key error detected
   1: Key information failed to load into key registers
   KERF is triggered upon any of the following errors:
   – CRYP_KxR/LR register write does not respect the correct order (refer to
     Section 37.4.16: CRYP key registers for details).
   – CRYP fails to load the key shared by SAES peripheral (KMOD = 0x2).
   KERF must be cleared by the application software, otherwise KEYVALID cannot be set. It
   can be done through IPRST bit of CRYP_CR, or when a correct key writing sequence starts.

Bit 5  Reserved, must be kept at reset value.

Bit 4  BUSY: Busy bit
   This flag indicates whether CRYP is idle or busy.
   0: Idle
   1: Busy
   CRYP is flagged as idle when disabled (CRYPEN = 0) or when the AES core is not
   processing any data. It happens when the last processing has completed, or CRYP is waiting
   for enough data in the input FIFO or enough free space in the output FIFO (that is in each
   case at least 4 words).
   CRYP is flagged as busy when processing a block data, preparing a key (ECB or CBC
   decryption only), or transferring a shared key from SAES peripheral.

Bit 3  OFFU: Output FIFO full flag
   0: Output FIFO is not full
   1: Output FIFO is full

Bit 2  OFNE: Output FIFO not empty flag
   0: Output FIFO is empty
   1: Output FIFO is not empty

Bit 1  IFNF: Input FIFO not full flag
   0: Input FIFO is full
   1: Input FIFO is not full

Bit 0  IFEM: Input FIFO empty flag
   0: Input FIFO is not empty
   1: Input FIFO is empty
37.8.3 CRYP data input register (CRYP_DINR)

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>DIN[31:16]</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Bits 31:0 DIN[31:0]: Data input
A four-fold sequential write to this bitfield during the Input phase results in pushing a complete 16-byte block into the CRYP input FIFO. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0].
Input FIFO can receive up to two 16-byte blocks of plaintext (when encrypting) or ciphertext (when decrypting).
If EN bit is set in CRYP_CR register, when at least four 32-bit words have been pushed into the input FIFO, and when at least four 32-bit words are free in the output FIFO, the CRYP automatically starts an encryption or decryption process, setting the BUSY bit.
Reading this register pops data off the input FIFO (oldest value is returned). The data present in the input FIFO are returned only if CRYPEN is cleared (undefined value is returned otherwise). Following one or more reads the FIFO must be flushed (setting the FFLUSH bit) prior to processing new data.

37.8.4 CRYP data output register (CRYP_DOUTR)

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>DOUT[31:16]</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DOUT[15:0]</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
<th>r</th>
</tr>
</thead>
</table>

Bits 31:0 DOUT[31:0]: Data output
A four-fold sequential read to this bitfield during the output phase results in retrieving a complete 16-byte block from the CRYP output FIFO. From the first to the fourth read, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0].
Output FIFO can store up to two 16-byte blocks of plaintext (when decrypting) or ciphertext (when encrypting).
When the output FIFO is empty a read returns an undefined value. Writes are ignored.
37.8.5  CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

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

Bit 1  **DOEN**: DMA output enable
- When this bit is set, DMA requests are automatically generated by the peripheral during the output data phase.
- 0: Outgoing data transfer from CRYP via DMA is disabled
- 1: Outgoing data transfer from CRYP via DMA is enabled

Bit 0  **DIEN**: DMA input enable
- When this bit is set, DMA requests are automatically generated by the peripheral during the input data phase.
- 0: Incoming data transfer to CRYP via DMA is disabled
- 1: Incoming data transfer to CRYP via DMA is enabled

37.8.6  CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

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

Bit 1  **OUTIM**: Output FIFO service interrupt mask
- This bit enables or disables (masks) the CRYP output FIFO service interrupt generation when OUTRIS is set.
- 0: Output FIFO interrupt is disabled (masked), masked interrupt status (OUTMIS) stays cleared
- 1: Output FIFO interrupt is enabled (not masked)
Bit 0  **INIM**: Input FIFO service interrupt mask  
This bit enables or disables (masks) the CRYP input FIFO service interrupt generation when INRIS is set.  
0: Input FIFO interrupt is disabled (masked), masked interrupt status (INMIS) stays cleared  
1: Input FIFO interrupt is enabled (not masked)

### 37.8.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18  
Reset value: 0x0000 0001

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

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

Bit 1  **OUTRIS**: Output FIFO service raw interrupt status  
This read-only bit is set by hardware when an output FIFO flag (OFFU or OFNE) is set in CRYP_SR register, regardless of the OUTIM mask bit value in CRYP_IMSCR register.  
0: No output FIFO event detected  
1: Output FIFO full or not empty detected; an interrupt is generated if OUTIM bit is set in CRYP_IMSCR register

Bit 0  **INRIS**: Input FIFO service raw interrupt status  
This read-only bit is set by hardware when an input FIFO flag (IFNF or IFEM) is set in CRYP_SR register, regardless of the INIM mask bit value in CRYP_IMSCR register.  
0: No input FIFO event detected  
1: Input FIFO empty or not full detected; an interrupt is generated if CRYPEN is set and if INIM bit is set in CRYP_IMSCR register

### 37.8.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C  
Reset value: 0x0000 0000

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

Bits 31:2  Reserved, must be kept at reset value.
Bit 1 **OUTMIS**: Output FIFO service masked interrupt status

This read-only bit is set by hardware when an output FIFO flag (OFFU or OFNE) is set in CRYP_SR register. If the OUTIM mask bit is cleared in CRYP_IMSCR register, the OUTMIS bit stays cleared (masked).

0: No output FIFO event detected or OUTIM mask cleared in CRYP_IMSCR
1: Output FIFO full or not empty detected, with an interrupt pending

The OUTMIS bit is cleared by reading data from the output FIFO until OFNE flag is cleared (output FIFO empty). It is not cleared by disabling CRYP with CRYPEN bit.

Bit 0 **INMIS**: Input FIFO service masked interrupt status

This read-only bit is set by hardware when an input FIFO flag (IFNF or IFEM) is set in CRYP_SR register. If the INIM mask bit is cleared in CRYP_IMSCR register, the INMIS bit stays cleared (masked).

0: No input FIFO event detected or INIM mask cleared in CRYP_IMSCR or CRYPEN bit cleared.
1: Input FIFO empty or not full detected, with an interrupt pending

The INMIS bit is cleared by writing data to the input FIFO until IFEM flag is cleared (there is at least one word in input FIFO), or by clearing CRYPEN.

When CRYP is disabled, INMIS bit stays low even if the input FIFO is empty.

**37.8.9 CYP key register 0L (CRYP_K0LR)**

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>K[255:240]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>K[239:224]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
</tbody>
</table>

 Bits 31:0 **K[255:224]**: Key bit x (x = 255 to 224)

This write-only bitfield contains the bits [255:224] of the AES encryption or decryption key, depending on the operating mode.

Write to CRYP_KxR/LR registers is ignored when CRYP is busy (BUSY bit set). When key is coming from the SAES peripheral (KMOD[1:0] = 0x2), write is also ignored. With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to CRYP_KxR/LR register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the CRYP_SR register.
### 37.8.10 CRYP key register 0R (CRYP_K0RR)

Address offset: 0x24  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **K[223:192]**: Key bit x (x = 223 to 192)  
This write-only bitfield contains the bits [223:192] of the AES encryption or decryption key, depending on the operating mode.  
Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

### 37.8.11 CRYP key register 1L (CRYP_K1LR)

Address offset: 0x28  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **K[191:160]**: Key bit x (x = 191 to 160)  
This write-only bitfield contains the bits [191:160] of the AES encryption or decryption key, depending on the operating mode.  
Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

### 37.8.12 CRYP key register 1R (CRYP_K1RR)

Address offset: 0x2C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **K[159:144]**: Key bit x (x = 159 to 144)  
This write-only bitfield contains the bits [159:144] of the AES encryption or decryption key, depending on the operating mode.  
Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.
37.8.13 CRYP key register 2L (CRYP_K2LR)

Address offset: 0x30
Reset value: 0x0000 0000

Bits 31:0 \( K[159:128] \): Key bit x (x = 159 to 128)
This write-only bitfield contains the bits [159:128] of the AES encryption or decryption key, depending on the operating mode.
Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

37.8.14 CRYP key register 2R (CRYP_K2RR)

Address offset: 0x34
Reset value: 0x0000 0000

Bits 31:0 \( K[127:96] \): Key bit x (x = 127 to 96)
This write-only bitfield contains the bits [127:96] of the AES encryption or decryption key, depending on the operating mode.
Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.
### 37.8.15 CRYP key register 3L (CRYP_K3LR)

Address offset: 0x38
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>K[63:48]</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **K[63:32]**: Key bit x (x = 63 to 32)
This write-only bitfield contains the bits [63:32] of the AES encryption or decryption key, depending on the operating mode. Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

### 37.8.16 CRYP key register 3R (CRYP_K3RR)

Address offset: 0x3C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>K[31:16]</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **K[31:0]**: Key bit x (x = 31 to 0)
This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode. Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

### 37.8.17 CRYP initialization vector register 0L (CRYP_IV0LR)

Address offset: 0x40
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>IV[127:112]</td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **IV[127:112]**: Initialization vector bits
This read/write bitfield contains the 16 initialization vector bits for the AES encryption or decryption operation, depending on the operating mode. Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.
Bits 31:0 **IVI[127:96]**: Initialization vector bit x (x = 127 to 96)  
This bitfield stores the initialization vector bits [127:96] for AES chaining modes other than ECB.  
The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).  
Write to this register is ignored when CRYP is busy (BUSY bit set).

### 37.8.18 CRYP initialization vector register 0R (CRYP_IV0RR)

Address offset: 0x44  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**IVI[95:80]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:0 **IVI[95:64]**: Initialization vector bit x (x = 95 to 64)  
This bitfield stores the initialization vector bits [95:64] for AES chaining modes other than ECB.  
The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).  
Write to this register is ignored when CRYP is busy (BUSY bit set).

### 37.8.19 CRYP initialization vector register 1L (CRYP_IV1LR)

Address offset: 0x48  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**IVI[63:48]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:0 **IVI[63:32]**: Initialization vector bit x (x = 63 to 32)  
This bitfield stores the initialization vector bits [63:32] for AES chaining modes other than ECB.  
The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).  
Write to this register is ignored when CRYP is busy (BUSY bit set).
37.8.20 **CRYP initialization vector register 1R (CRYP IV1RR)**

Address offset: 0x4C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>IV[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **IV[31:0]**: Initialization vector bit x (x = 31 to 0)
This bitfield stores the initialization vector bits [31:0] for AES chaining modes other than ECB.
The value stored in CRYP IVxR/LR registers is updated by hardware after each computation
round (when applicable).
Write to this register is ignored when CRYP is busy (BUSY bit set).

37.8.21 **CRYP context swap GCM-CCM registers (CRYP CSGCMCCMxR)**

Address offset: 0x50 + 0x4 * x, (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSGCMCCM[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSGCMCCM[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **CSGCMCCM[31:0]**: Context swap for GCM/GMAC and CCM modes
CRYP CSGCMCCMxR registers contain the complete internal register states of the CRYP
when the GCM, GMAC or CCM processing of the current task is suspended to process a
higher-priority task. Refer to Section 37.4.8: CRYP suspend and resume operations for more
details.
CRYP CSGCMCCMxR registers are not used in other chaining modes than GCM, GMAC or
CCM.
37.8.22 **CRYP context swap GCM registers (CRYP_CSGCMxR)**

Address offset: 0x70 + 0x4 * x, (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Address offset</th>
<th>CRYP_CSGCMxR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRYP_CR</td>
</tr>
<tr>
<td>0x04</td>
<td>CRYP_SR</td>
</tr>
<tr>
<td>0x08</td>
<td>CRYP_DINR</td>
</tr>
<tr>
<td>0x0C</td>
<td>CRYP_DOUTR</td>
</tr>
<tr>
<td>0x10</td>
<td>CRYP_DMCR</td>
</tr>
<tr>
<td>0x14</td>
<td>CRYP_IMSCR</td>
</tr>
</tbody>
</table>

Bits 31:0 **CSGCM[31:0]**: Context swap for GCM/GMAC modes
CRYP_CSGCMxR registers contain the complete internal register states of the CRYP when the GCM or GMAC processing of the current task is suspended to process a higher-priority task. Refer to Section 37.4.8: CRYP suspend and resume operations for more details. CRYP_CSGCMxR registers are not used in other chaining modes than GCM or GMAC.

### 37.8.23 CRYP register map

Table 350. CRYP register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CRYP_CR</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x04</td>
<td>CRYP_SR</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x08</td>
<td>CRYP_DINR</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x0C</td>
<td>CRYP_DOUTR</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x10</td>
<td>CRYP_DMCR</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x14</td>
<td>CRYP_IMSCR</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
Table 350. CRYP register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>( \text{K}[255:224] )</th>
<th>( \text{K}[223:192] )</th>
<th>( \text{IVI}[127:96] )</th>
<th>( \text{IVI}[95:64] )</th>
<th>( \text{IVI}[63:32] )</th>
<th>( \text{IVI}[31:0] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x18</td>
<td>CRYP_RISR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x1C</td>
<td>CRYP_MISR</td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x20</td>
<td>CRYP_K0LR</td>
<td>( \text{K}[255:224] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x24</td>
<td>CRYP_K0RR</td>
<td>( \text{K}[223:192] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x38</td>
<td>CRYP_K3LR</td>
<td>( \text{K}[63:32] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x3C</td>
<td>CRYP_K3RR</td>
<td>( \text{K}[31:0] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x40</td>
<td>CRYP_IV0LR</td>
<td>( \text{IVI}[127:96] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x44</td>
<td>CRYP_IV0RR</td>
<td>( \text{IVI}[95:64] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x48</td>
<td>CRYP_IV1LR</td>
<td>( \text{IVI}[63:32] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x4C</td>
<td>CRYP_IV1RR</td>
<td>( \text{IVI}[31:0] )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x50 + ( x \times 0x4 ), last address 0x6C</td>
<td>CRYP_CSGMCCMxR</td>
<td>CSGMCCMx[31:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x70 + ( x \times 0x4 ), last address 0x9C</td>
<td>CRYP_CSGCMxR</td>
<td>CSGCMx[31:0]</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
38 Hash processor (HASH)

38.1 Introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than $2^{64}$ bits (for SHA-1, SHA-224 and SHA-256) or less than $2^{128}$ bits (for SHA-384, SHA-512).

38.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, Secure Hash Standard (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, Digital Signature Standard (DSS)
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO, corresponding to a 64-byte block size (16 x 32 bits)
- Automatic padding to complete the input bit string to fit digest minimum block size
- AHB slave peripheral, accessible by 32-bit words only (else an AHB error is generated)
- $8 \times 32$-bit words (H0 to H15) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers
  - Hashing computation suspend/resume mechanism, including DMA
38.3 HASH implementation
The devices have a single instance of HASH peripheral.

38.4 HASH functional description

38.4.1 HASH block diagram
*Figure 389* shows the block diagram of the hash processor.

**Figure 389. HASH block diagram**

38.4.2 HASH internal signals
*Table 351* describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

**Table 351. HASH internal input/output signals**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hash_hclk</td>
<td>digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>hash_it</td>
<td>digital output</td>
<td>Hash processor global interrupt request</td>
</tr>
<tr>
<td>hash_dma</td>
<td>digital input/output</td>
<td>DMA burst request/ acknowledge</td>
</tr>
</tbody>
</table>
38.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see Table 352).

Table 352. Information on supported hash algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Message digest size (in bits)</th>
<th>Block size (in bytes)(1)</th>
<th>Message length</th>
<th>Bit string message</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>160</td>
<td>64</td>
<td>&lt; 2^{64} bits</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA2-224</td>
<td>224</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA2-256</td>
<td>256</td>
<td>64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SHA2-384</td>
<td>384</td>
<td>128</td>
<td>&lt; 2^{128} bits</td>
<td>Yes</td>
</tr>
<tr>
<td>SHA2-512</td>
<td>512(2)</td>
<td>128</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017). Any change to a message in transit results, with very high probability, in a different message digest, and the signature fails to verify.

38.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string "abc" with a bit string representation of "01100001 01100010 01100011" is represented by a 32-bit word \texttt{0x00636261}, and 8-bit words \texttt{0x61626300}.

Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be
performed on data from the input FIFO before entering the little-endian hash processing core. Figure 390 shows how the hash processing core 32-bit data block \( M_0 \ldots 31 \) is constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR).

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

**Figure 390. Message data swapping feature**
38.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in Table 352: Information on supported hash algorithms.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in Section 38.4.4: Message data feeding, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals \((\text{NBWE}-1) \times 4\). This step is not required if the block size is already known (see Table 352 for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to Section 38.4.7: HMAC operation for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.

2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:
     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.
     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the
block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see Section 38.4.6 for details).

– Data are filled as a single DMA transfer (MDMAT = 0):
Partial digest computations are triggered automatically each time the FIFO is full. The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see Section 38.4.6 for details).

– Data are filled using multiple DMA transfers (MDMAT = 1):
Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in Table 353.

Table 353. Hash processor outputs

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Valid output registers</th>
<th>Most significant bit</th>
<th>Digest size (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>HASH_H0 to HASH_H4</td>
<td>HASH_H0[31]</td>
<td>160</td>
</tr>
<tr>
<td>SHA2-224</td>
<td>HASH_H0 to HASH_H6</td>
<td>HASH_H0[31]</td>
<td>224</td>
</tr>
<tr>
<td>SHA2-256</td>
<td>HASH_H0 to HASH_H7</td>
<td>HASH_H0[31]</td>
<td>256</td>
</tr>
<tr>
<td>SHA2-384</td>
<td>HASH_H0 to HASH_H11</td>
<td>HASH_H0[31]</td>
<td>384</td>
</tr>
<tr>
<td>SHA2-512</td>
<td>HASH_H0 to HASH_H15</td>
<td>HASH_H0[31]</td>
<td>512(1)</td>
</tr>
</tbody>
</table>

1. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to Section 38.4.7: HMAC operation.

38.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.
Padding processing
Detailed padding sequences, with DMA enabled or disabled, are described in Section 38.4.5: Message digest computing.

Padding example
As specified by Federal Information Processing Standards PUB 180-4, message padding consists in appending a “1” followed by \( k \) “0”s, itself followed by a 64-bit integer that is equal to the length \( L \) in bits of the message. These three padding operations generate a padded message of length \( L + 1 + k + 64 \), which by construction is a multiple of 512 bits.

For the hash processor, the “1” is added to the last word written to the HASH_DIN register at the bit position defined by the NBLW bitfield, and the remaining upper bits are cleared (“0”s).

Example from FIPS PUB180-4
Let us assume that the original message is the ASCII binary-coded form of "abc", of length \( L = 24 \):

\[
\begin{array}{cccc}
\text{byte 0} & \text{byte 1} & \text{byte 2} & \text{byte 3} \\
01100001 & 01100010 & 01100011 & \text{UUUUUUU} \\
& \text{--- 1st word written to HASH_DIN ---} & \\
\end{array}
\]

NBLW has to be loaded with the value 24: a “1” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

\[
01100001 \ 01100010 \ 01100011 \ \text{UUUUUUU}
\]

Since \( L = 24 \), the number of bits in the above bit string is 25, and 423 “0” bits are appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

\[
\begin{array}{cccc}
616263 & 80 & 0000000 & 0000000 \\
& 0000000 & 0000000 & 0000000 \\
& 0000000 & 0000000 & 0000000 \\
& 0000000 & 0000000 \\
\end{array}
\]

The message length value, \( L \), in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

\[
\begin{array}{cccc}
61626380 & 00000000 & 00000000 & 00000000 \\
& 00000000 & 00000000 & 00000000 \\
& 00000000 & 00000000 & 00000000 \\
& 00000000 & 00000000 & 00000018 \\
\end{array}
\]
If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where 'U' means don’t care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x10** is written to the HASH_STR register to start the message padding (described above) and then perform the digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

   - HASH_HR0 = 0xA9993E36
   - HASH_HR1 = 0x4706816A
   - HASH_HR2 = 0xBA3E2571
   - HASH_HR3 = 0x7850C26C
   - HASH_HR4 = 0x9CD0D89D

### 38.4.7 HMAC operation

#### Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

\[
\text{HMAC(message)} = \text{Hash((Key | pad) XOR opad | Hash((Key | pad) XOR ipad | message))}
\]

Where:

- **opad** = [0x5C]_n (outer pad) and **ipad** = [0x36]_n (inner pad)
- \([X]_n\) represents a repetition of \(X\) \(n\) times, where \(n\) equal to the byte size of the underlying hash function data block (\(n = 64\) when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length \(n\) defined above. If the key length is greater than \(n\), the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

**Note:** *HMAC mode of the hash processor can be used with all supported algorithms.*

#### HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into
HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.

3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in Section 38.4.5: Message digest computing.

4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 38.4.11: HASH processing time.

Endianness management details can be found in Section 38.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of “Sample message for keylen = blocklen”, of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. Inner hash key input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617 18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F 30313233 34353637 38393A3B 3C3D3E3F

2. Message input (length = 34, that is, padding required). HASH_STR must be set to 0x20 to start message padding and inner hash computation (see ‘U’ as don’t care)

   53616D70 6C65206D 65737361 67652066 6F722065 6E3D626C 656EUUUU

3. Outer hash key input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.

4. Final outer hash computing is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

   HASH_HR0 = 0xF5FD596EE
   HASH_HR1 = 0x78D5555C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
38.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in Figure 391.

![Figure 391. HASH suspend/resume mechanism](image)

To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.
Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see Figure 391).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.
To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

### 38.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to Section 38.4.5: Message digest computing.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

\[ NBLW = \text{Len}(\text{Message}) \mod 32 \]

where \( x \mod 32 \) gives the remainder of \( x \) divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

*Note:* No interrupt is associated to DMAS bit. When MDMAT is set, the size of the transfer must be a multiple of four words.

### 38.4.10 HASH error management

No error flags are generated by the hash processor.

### 38.4.11 HASH processing time

*Table 354* summarizes the time required to process an intermediate block for each mode of operation.

<table>
<thead>
<tr>
<th>Mode of operation</th>
<th>Block size (in bytes)</th>
<th>FIFO load&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Computation phase</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>64</td>
<td>16</td>
<td>66</td>
<td>82</td>
</tr>
<tr>
<td>SHA2-224</td>
<td></td>
<td>16</td>
<td>16</td>
<td>66</td>
</tr>
<tr>
<td>SHA2-256</td>
<td></td>
<td>32</td>
<td>50</td>
<td>82</td>
</tr>
<tr>
<td>SHA2-384</td>
<td></td>
<td>32</td>
<td>66</td>
<td>98</td>
</tr>
<tr>
<td>SHA2-512&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>128</td>
<td>32</td>
<td>66</td>
<td>98</td>
</tr>
</tbody>
</table>

---

<sup>(1)</sup> Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- 1 to 2.5 for a hash message
- ~2.5 for an HMAC input-key
- 1 to 2.5 for an HMAC message
- ~2.5 for an HMAC output key in case of a short key
- 3.5 to 5 for an HMAC output key in case of a long key

38.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. Table 355 gives a summary of the available features.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASH</td>
<td>Digest computation completed</td>
<td>DCIS</td>
<td>DCIE</td>
<td>Clear DCIS or set INIT</td>
</tr>
<tr>
<td></td>
<td>Data input buffer ready to get a new block</td>
<td>DINIS</td>
<td>DINIE</td>
<td>Clear DINIS or write to HASH_DIN</td>
</tr>
</tbody>
</table>

38.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

38.6.1 HASH control register (HASH_CR)

Address offset: 0x000
Reset value: 0x0000 0000
### Hash processor (HASH)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

 Bits 31:21: Reserved, must be kept at reset value.

 Bits 20:17 **ALGO[3:0]:** Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect.

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1
0001: reserved
0010: SHA2-224
0011: SHA2-256
1100: SHA2-384
1101: SHA2-512/224
1110: SHA2-512/256
1111: SHA2-512

 Bit 16 **LKEY:** Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see Table 352: Information on supported hash algorithms for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.
1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

 Bit 15: Reserved, must be kept at reset value.

 Bit 14: Reserved, must be kept at reset value.

 Bit 13 **MDMAT:** Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.
1: DCAL is not automatically set at the end of a DMA transfer.

 Bit 12 **DINNE:** DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit. This bit is read-only.

 Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield. This bit is read-only.

 Bit 7: Reserved, must be kept at reset value.
38.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004
Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see Table 352: Information on supported hash algorithms for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.
When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

**Note:** When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

### Bits 31:0 DATAIN[31:0]: Data input

- Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.
- Reading this register returns zeros.

### 38.6.3 HASH start register (HASH_STR)

- **Address offset:** 0x008
- **Reset value:** 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.
38.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result.

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

**Note:** When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).
**HASH aliased digest register x (HASH_HRAx)**

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hx[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>Hx[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **Hx[31:0]**: Hash data x

Refer to *Section 38.6.4: HASH digest registers* introduction.

**HASH digest register x (HASH_HRx)**

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hx[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>Hx[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **Hx[31:0]**: Hash data x

Refer to *Section 38.6.4: HASH digest registers* introduction.

**HASH supplementary digest register x (HASH_HRx)**

Address offset: 0x310 + 0x4 * x, (x = 5 to 15)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hx[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>Hx[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **Hx[31:0]**: Hash data x

Refer to *Section 38.6.4: HASH digest registers* introduction.
38.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:2</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 1</td>
<td>DCIE: Digest calculation completion interrupt enable</td>
<td>0: Digest calculation completion interrupt disabled, 1: Digest calculation completion interrupt enabled.</td>
</tr>
<tr>
<td>Bit 0</td>
<td>DINIE: Data input interrupt enable</td>
<td>0: Data input interrupt disabled, 1: Data input interrupt enabled</td>
</tr>
</tbody>
</table>

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

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

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

38.6.6 HASH status register (HASH_SR)

Address offset: 0x024
Reset value: 0x0011 0001

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>Bit 15</td>
<td>DINNE: DIN not empty</td>
<td>This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set. 0: No data are present in the data input buffer, 1: The input buffer contains at least one word of data</td>
</tr>
<tr>
<td>Bit 14</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

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

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 NBWE[4:0]: Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 DINNE: DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data
Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP - block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)
1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free).

It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

38.6.7 **HASH context swap registers**

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.
HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)
Reset value: 0x0022 0002 (HASH_CSR0)
Reset value: 0x0020 0000 (HASH_CSR2)
Reset value: 0x0000 0000 (others)

38.6.8 HASH register map

Table 356. HASH1 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Bits 31:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>HASH_CR</td>
<td></td>
<td>CSx[31:16]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>HASH_DIN</td>
<td>DATAIN[31:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>HASH_STR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>HASH_HRAx</td>
<td>Hx[31:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>HASH_IMR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>HASH_SR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td></td>
</tr>
</tbody>
</table>

Reserved
Refer to Section 2.3 on page 149 for the register boundary addresses.
39 Memory cipher engine (MCE)

39.1 Introduction

Memory cipher engine (MCE) defines, in a given address space, multiple regions with specific security setup (encryption, privilege, write). All system bus traffic going through an encrypted region is managed on-the-fly by the MCE, automatically decrypting reads and encrypting writes if authorized.

Multiple ciphering option (stream, block, fast block) are available to offer the best security versus performance trade-off.

Note: When MCE is used in conjunction with XSPI it is mandatory to access the flash memory using the memory map mode of the flash memory controller.

39.2 MCE main features

- System bus in-line encryption (for writes) and decryption (for reads), based on embedded firewall programming
  - Four encryption modes per region (maximum 4 regions available): no encryption (bypass mode), stream cipher, block cipher and fast block cipher modes
  - Start and end of regions defined with 4-Kbytes granularity
  - Default filtering (region 0): any access granted
  - Region 1 to 4 access filtering criteria: privilege, write
- Supported block ciphers: AES or Noekeon (12 round version)
- Supported chaining modes: block and stream
  - Block mode with AES cipher is compatible with ECB mode specified in NIST FIPS publication 197 Advanced encryption standard (AES) (normal or fast).
  - Stream mode with AES cipher is compliant with CTR mode specified in NIST SP800-38A Recommendation for Block Cipher Modes of Operation.
  - Includes a leakage resilient mode of operation as defense against side channel attacks (SCA).
- One set of write-only and lockable master key registers per block cipher (normal, fast)
- Two sets of lockable cipher contexts (128-bit key, IV), usable for stream and block ciphers
- Optimization for XSPI data pre-fetching mechanism (stream cipher only)
- Read-write arbitration scheme, for better read performances
- AHB configuration port, privileged aware
- AXI system bus master/slave interfaces (64-bit)
  - Support for any AXI-64bit read transactions
  - When encryption is enabled, support for AXI-64bit INCRx (x=1 to 8) and WRAPx (x=4) write transactions
39.3 MCE implementation

This manual describes the full set of features implemented in MCE peripheral. In this device MCE1, MCE2 and MCE3 instances implement the set of features defined on Table 357.

Table 357. MCE implementation

<table>
<thead>
<tr>
<th>MCE features</th>
<th>MCE1</th>
<th>MCE2 and MCE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of regions</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>Cipher engines</td>
<td>AES x2</td>
<td>12 rounds Noekeon x2</td>
</tr>
<tr>
<td>Derive key function</td>
<td>normal, fast</td>
<td></td>
</tr>
<tr>
<td>Master key</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Chaining modes (encryption mode)</td>
<td>block, stream</td>
<td>block</td>
</tr>
<tr>
<td>Cipher context(s)</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

MCE peripheral generates interrupts to the Cortex® CPU interrupt controller (mce_it).
39.4 MCE functional description

39.4.1 MCE block diagram

*Figure 392* shows the memory cipher engine block diagram, associated with one or more memory interface peripheral.

![MCE block diagram](image)

1. Refer to *Section 39.3: MCE implementation*.

39.4.2 MCE internal signals

*Table 358* describes the user relevant internal signals interfacing the MCE peripheral.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type(1)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mce_sck</td>
<td>input</td>
<td>AXI input clock</td>
</tr>
<tr>
<td>mce_hck</td>
<td>input</td>
<td>AHB input clock for MCE</td>
</tr>
<tr>
<td>mce_it</td>
<td>output</td>
<td>MCE interrupt request</td>
</tr>
</tbody>
</table>

1. All digital.
39.4.3 MCE programming

Overview

Through MCE registers application software defines security regions, as shown on Figure 393.

Figure 393. MCE region programming

1. Covers all addressable space.

There are three types of accesses filtered by MCE:

1. Access to an area located in one enabled region (accesses 1). In this case region filtering rules apply. If encryption is enabled cipher engine is used accordingly.
2. Access to an area overlapping two enabled regions (access 2):
   - If one region is encrypted, this region defines the write and privilege accesses, as well as the encryption.
   - If both regions are encrypted, access is automatically rejected with an error.
   - If both regions are not encrypted, lowest index region defines the write and privilege accesses.

Access to an area not belonging to any enabled region (accesses 3). In this case primary region 0 filtering applies (any access granted, no encryption).

Configuring regions

When application needs to use any region x following sequence can be used, for example at boot time.

1. Application sets a start and end address for each of the required region x, using MCE_SADDRx and MCE_EADDRx registers.
2. For each of the above region x application programs MCE_ATTRx, granting write access to the memory or not.
3. Application finalizes the programming by configuring MCE_REGCRx setting up security parameters (encryption, privilege). Initializing encryption is detailed in Section 39.4.5 and Section 39.4.5.
4. After final check on above configuration application enables each required region z, setting BREN bit in MCE_REGCRz.
MCE configuration can be fixed until next SoC reset when GLOCK is set in MCE_CR register.

Above application must be privileged if PRIV bit is set in register MCE_PRIVCFGR.

**Note:** When ENCDIS bit is set in MCE_SR, if ENC bit and BREN bits are set in MCE_REGCRx all writes to this region x are ignored and all read requests return zero. See Section 39.4.8 for details.

**Note:** When an MCE instance uses at least one crypto context, to avoid issues at runtime it is recommended to verify in MCE_CCxCFGR registers that, each time CCEN=1, KEYCRC is different from zero. After this check it is recommended to set GLOCK in MCE_CR to freeze the MCE configuration.

### Runtime modification of region configuration

While it is possible to update on-the-fly the enabled region access filtering (write, privilege), changes to the following configurations are ignored when BREN bit is set in MCE_REGCRx register:

- MCE_SADDRx and MCE_EADDRx address registers
- ENC and CTXID bitfields in MCE_REGCx register

**Note:** When BREN is cleared primary region 0 access rules applies instead of base region rules (see Overview).

GLOCK bit should be used to write lock the following registers: MCE_MKEYRx, MCE_FMKEYRx, MCE_CCxCFGR, MCE_CCxNR0/1 and MCE_CCxKEYR, in order to avoid unexpected changes to the cipher keys/context while used by the MCE.

### Encrypting with MCE

MCE automatically encrypts on-the-fly any granted write transaction to a region with encryption properly initialized. When stream cipher is selected it is important to activate write protection as soon as the whole region has been encrypted (read-only region).

When MCE is used in conjunction with XSPI it is mandatory to set the Flash memory controller in memory map mode. It is also required to use the DMA to perform writes to the flash memory using 16 bytes bursts.

### 39.4.4 MCE reset and clocks

MCE configuration port is clocked by the AHB bus clock.

MCE system bus interface is clocked by the AXI clock. AHB interface does not need to be clocked for the system bus to be usable.

When MCE is kept under reset, no traffic can go through the system bus. Out of reset default firewall configuration applies, as described in Section 39.4.3.

The clocks of MCE peripheral and its associated memory controller are enabled together in the RCC, and the MCE peripheral is reset by a system reset or when associated memory controller is reset (see Section 7: Reset and clock control (RCC)).

### 39.4.5 MCE block cipher encryption mode

**Caution:** When MCE is used in conjunction with XSPI it is mandatory to read or write the flash memory using the memory map mode of the flash memory controller.
Figure 394 details the block cipher implementations in the MCE peripheral. Top figure is a standard encryption (for writes), bottom figure is a fast decryption (for reads).

**Figure 394. MCE implementation of block ciphers**

1. Operation starts from the capture of the address.
2. Operation starts from availability of data input.
3. Only required for AES cipher.

Address \( i \) is most significant bits of physical address of block \( i \) + block \( i+1 \). Result of key derivation primitive can be used for two consecutive 16-byte data words. Since the normal key derivation function is leakage resilient the master key information is protected against side channel attacks (SCA).

**MASTER_KEY** must be stored in write-only MCE_MKEYRx registers.
**FAST_MASTER_KEY** must be stored in write-only MCE_FMKEYRx registers. Alternatively, MCE_CCzKEYRx registers can be used if cipher context \( z \) is selected in MCE_REGCR (CTXID=z).

Each time a key material has been successfully written in MCE, the following flags are set:
- MK VALID bit in MCE_SR is set when a complete key has been written in MCE_MKEYR registers.
- FMK VALID bit in MCE_SR is set when a complete key has been written in MCE_FMKEYR registers.
- KEYCRC[7:0] in MCE_CCzCFGR is loaded with 8-bit checksum when a complete key has been written in MCE_CCzKEYR registers. See KEYCRC bitfield for details.
Bypass mode is selected instead of cipher mode if the selected key is not marked as valid with above flags. See ENC bitfield description for details.

**Note:** With AES block cipher, writing (encryption) requires extra cycles in order to perform the required key scheduling to obtain the encryption key. See next table for details.

Fast key derivation function may be sensitive against side channel attacks. Hence master key used in normal mode is never used in fast mode.

Relative performances are described on Table 359. Refer to Section 39.3: MCE implementation to identify which cipher is used for which MCE instance.

### Table 359. MCE block cipher latencies

<table>
<thead>
<tr>
<th>Cipher type</th>
<th>Block mode</th>
<th>ENC bit in MCE_REGCRx</th>
<th>Latency in AXI cycles</th>
<th>Optimization for sequential accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>for 16 bytes data</td>
<td>for 32 bytes data</td>
</tr>
<tr>
<td>AES(1)</td>
<td>Normal(2)</td>
<td>10</td>
<td>14+11= 25</td>
<td>14+11x2= 36</td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>11</td>
<td>4+11= 15</td>
<td>4+11x2= 26</td>
</tr>
<tr>
<td>Noekeon</td>
<td>Normal</td>
<td>10</td>
<td>14+7= 21</td>
<td>14+7x2= 28</td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>11</td>
<td>4+7= 11</td>
<td>4+7x2= 18</td>
</tr>
</tbody>
</table>

1. Additional 10 cycles are required when performing encryption/ writes
2. Leakage resilient mode of operation as defense against side channel attacks.

### 39.4.6 MCE stream cipher encryption mode

*Figure 395* details the stream cipher implementations in the MCE peripheral. It pictures encryptions for AXI writes.

### Figure 395. MCE implementation of stream cipher

1. Operation starts from the capture of the address.

Every 128-bit data block (i), a counter (i) information is computed, as defined below:

- Counter[127:0] = MCE_CCzNR1[31:0] || MCE_CCzNR0[31:0] || 0b0000000000000000 || MCE_CCzCFGR[31:16] || 0b0000 || Address_i [31:4].
Address \(_i\) is the physical address of block \((i)\), modulo 16 bytes.

Version and nonce are stored by application respectively in MCE CCzCFGR and MCE CCzNReq registers, with cipher context \(z\) selected in MCE_REGCR (CTXID=\(z\)).

The least significant byte of Data_IN and Data_OUT are stored in the first 8 bits of the data blocks.

CC_KEY is stored in write-only MCE CCzKEYRx registers, with cipher context \(z\) selected in MCE_REGCR (CTXID=\(z\)). Bypass mode is selected instead of stream cipher mode if the selected key is not marked as valid, i.e. KEYCRC[7:0] in MCE_CCyCFGR is different from 0x0.

Note: Stream cipher mode offers no protection against bit-flip attacks, and is sensitive against side channel attacks.

Caution: In order to secure the confidentiality of the information encrypted with the stream cipher in region \(x\), it is important to clear the WREN bit in MCE_ATTRx register as soon as the whole region has been encrypted by the application (read-only region).

Caution: When MCE is used in conjunction with XSPI it is mandatory to read or write the flash memory using the memory map mode of the flash memory controller.

Relative performances are described on Table 360. Refer to Section 39.3: MCE implementation to identify which cipher is used for which MCE instance.

<table>
<thead>
<tr>
<th>Cipher type</th>
<th>Stream mode</th>
<th>ENC bit in MCE_REGCRx</th>
<th>Latency for 16 bytes data (in AXI cycles)</th>
<th>Optimization for sequential accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>Normal</td>
<td>01</td>
<td>11</td>
<td>Yes</td>
</tr>
</tbody>
</table>

39.4.7 MCE AXI traffic management

MCE includes a special AXI-64 read-write arbitration scheme, designed to speed-up multiple reads. More specifically, when no write is on-going in MCE the two cipher cores can be allocated to decrypt two reads in parallel. In this case an incoming write request is stalled until the cipher core allocated to writes is free.

Note: Best performances are obtained when transactions are aligned on two 64-bit words for regions where block cipher is selected. See Section 39.4.5 for details.

39.4.8 MCE encryption disable options

When MCE feature is not available in the product encryption/decryption feature of MCE is not usable, for any region. It means that reads (resp. writes) through MCE are not decrypted (resp. encrypted).

When an unexpected event occurs, the ENCDIS bit is set in MCE_SR register and the following effects apply:

- All previously written key materials are erased.
- If ENC bit and BREN bits are set in MCE_REGCRx, all writes to this region \(x\) are ignored and all read requests return zero.

In such an event, the application must reset the MCE peripheral to be able to use encrypted regions again.

Table 360. MCE stream cipher latencies
39.4.9 MCE error management

When an illegal access to a configuration register occurs (bad privilege) corresponding CAEF bit is set in MCE_IASR. Application cleans this flag using CAEF bit in MCE_IACR.

When an illegal access through the system bus occurs (bad privilege, forbidden write, overlapping encrypted regions) corresponding IAEF bit is set in MCE_IASR. Application cleans this flag using IAEF bit in MCE_IACR.

Additional debug information on system bus illegal accesses can be read in illegal access error status register (MCE_IAESR) and optionally the illegal address register (MCE_IADDR).

Illegal read (resp. write) through the system bus returns zero (resp. is ignored).

Read to write-only or write to read-only registers do not trigger an interrupt. Reading any MCE key register returns zeros.

39.5 MCE interrupts

There are two individual maskable interrupt sources generated by the MCE, signaling the following error events:

- Illegal access through the configuration bus (CAEF), see Section 39.4.9
- Illegal access through the system bus (IAEF), see Section 39.4.9

Those interrupt sources are connected to the same global interrupt request signal mce_it.

Setting the appropriate mask bit in MCE_IAIER register enables the interrupt. The status of the individual interrupt events can be read from MCE_IASR, and cleared in MCE_IACR register.

Table 361 gives a summary of the available features.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCE</td>
<td>Configuration access error</td>
<td>CAEF</td>
<td>CAEIE</td>
<td>Set CAEF bit in MCE_IACR</td>
</tr>
<tr>
<td></td>
<td>Illegal access error</td>
<td>IAEF</td>
<td>IAEIE</td>
<td>Set IAEF bit in MCE_IACR</td>
</tr>
</tbody>
</table>

1. Read in MCE_IASR register
39.6 MCE registers

39.6.1 MCE configuration register (MCE_CR)

Address offset: 0x000
Reset value: 0x0000 0000

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR.

Writes are ignored if GLOCK is set in MCE_CR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 1 **MKLOCK**: Master keys lock
Lock the master key configurations until next reset. This bit is cleared by default and once set it cannot be reset until MCE reset.
0: Writes to MCE_MKEYRx and MCE_FMKEYRx registers are allowed
1: Writes to MCE_MKEYRx and MCE_FMKEYRx registers are ignored until next MCE reset.
Effect of this bit depends on the number of master keys. See Section 39.3: MCE implementation for details.

Bit 0 **GLOCK**: Global lock
Lock the configuration of most MCE registers until next reset. This bit is cleared by default and once set it cannot be reset until MCE reset.
0: MCE registers are writable
1: All writes to MCE registers are ignored, with the exception of MCE_IACR and MCE_IAIER registers.

39.6.2 MCE status register (MCE_SR)

Address offset: 0x004
Reset value: 0x0000 0000

Privileged read access only if PRIV is set in MCE_PRIVCFGR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>
39.6.3 Memory cipher engine (MCE) illegal access status register (MCE_IASR)

Address offset: 0x008
Reset value: 0x0000 0000

Privileged read access only if PRIV is set in MCE_PRIVCFGR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IAEF</td>
<td>CAEF</td>
</tr>
</tbody>
</table>

- **Bit 31:2** Reserved, must be kept at reset value.

- **Bit 1** **IAEF**: Illegal access error flag
  - This bit is set when an illegal access is detected on the system bus. More details on the error can be found in MCE_IAESR and MCE_IADDR registers.
  - This bit is cleared by setting corresponding bit in MCE_IACR register.

- **Bit 0** **CAEF**: Configuration access error flag
  - This bit is set when an illegal access to any MCE configuration register is detected. Bit is cleared by setting corresponding bit in MCE_IACR register. No additional details on the error is available.
### 39.6.4 MCE illegal access clear register (MCE_IACR)

Address offset: 0x00C  
Reset value: 0x0000 0000  
Privileged write access only if PRIV is set in MCE_PRIVCFGR.

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 1 IAEF**: Illegal access error flag clear  
Set this bit to clear IAEF bit in MCE_IASR register. Clearing IAEF bit permits to capture new error information in MCE_IAESR and MCE_IADDR registers.  
Note that clearing this bit does not clear RISAB_IADDR register.

**Bit 0 CAEF**: Configuration access error flag clear  
Set this bit to clear CAEF bit in MCE_IASR register.

### 39.6.5 MCE illegal access interrupt enable register (MCE_IAIER)

Address offset: 0x010  
Reset value: 0x0000 0000  
Privileged read and write access only if PRIV is set in MCE_PRIVCFGR.

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 1 IAEIE**: Illegal access error interrupt enable  
0: Interrupt generation on illegal access errors is disabled  
1: Interrupt generation when an illegal access error occurs (IAEF=1)  

**Bit 0 CAEIE**: Configuration access error interrupt enable  
0: Interrupt generation on configuration access errors is disabled  
1: Interrupt generation when a configuration access error occurs (CAEF=1)
39.6.6 MCE privileged configuration register (MCE_PRIVCFGR)

Address offset: 0x1C
Reset value: 0x0000 0000
Privileged write access only.
Any read is allowed. Writes are ignored if GLOCK is set in MCE_CR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 0 PRIV: Privileged configuration
0: Privileged and unprivileged access are granted to MCE registers
1: Only privileged access are granted to MCE registers

39.6.7 MCE illegal access error status register (MCE_IAESR)

Address offset: 0x020
Reset value: 0x0000 0000
Privileged read access only if PRIV is set in MCE_PRIVCFGR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 7 IANRW: Illegal access read/write
When IAEF bit is set in MCE_IASR register IANRW bit captures the access type of the illegal access detected.
0: Illegal access was a data read or an instruction fetch.
1: Illegal access was a data write.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 IAPRIV: Illegal access privilege
When IAEF bit is set in MCE_IASR register IAPRIV bit captures the privileged state of the master that issued the illegal access detected on the AXI system bus.
0: Illegal access was unprivileged.
1: Illegal access was privileged.

Bits 3:0 Reserved, must be kept at reset value.
39.6.8 MCE illegal address register (MCE_IADDR)

Address offset: 0x024
Reset value: 0x0000 0000

Privileged read access only if PRIV is set in MCE_PRIVCFGR.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 IADD[31:0]: Illegal address
When IAEF bit is set in MCE_IASR register IADD bitfield captures the 32-bit bus address of the erroneous access. Additional information can be found in MCE_IAESR register.

39.6.9 MCE region x configuration register (MCE_REGCRx)

Address offset: 0x040 + 0x10 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR. Writes are ignored if GLOCK is set in MCE_CR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 16  PRIV: Privileged region
This bit is taken into account only if BREN is set.
0: Application can access to region x in privileged and unprivileged mode.
1: Application can access to region x in privileged mode only.

Bits 15:14  ENC[1:0]: Encrypted region
Those bits are taken into account only if BREN is set and if the corresponding encryption feature is available in the MCE instance (see Section 39.3: MCE implementation). Write to those bits is ignored if BREN is set.
00: No effects
01: Stream cipher - All allowed read (resp. write) requests are decrypted (resp. encrypted) using the stream cipher, when CCEN bit is set in MCE_CCzCFGR (where CTXID=z).
10: Block cipher - All allowed read (resp. write) requests are decrypted (resp. encrypted) using the SCA resistant block cipher. If CTXID=0 and MKVALID=0 bypass mode is selected instead. If CTXID=z and CCEN bit is cleared in MCE_CCzCFGR bypass mode is also selected.
11: Fast block cipher - All allowed read (resp. write) requests are decrypted (resp. encrypted) using the fast block cipher. If CTXID=0 and FMKVALID=0 bypass mode is selected instead. If CTXID=z and CCEN bit is cleared in MCE_CCzCFGR bypass mode is also selected.

Bits 13:11  Reserved, must be kept at reset value.

Bits 10:9  CTXID[1:0]: Context ID
This bitfield defines the cryptographic context used by the cipher engine assigned to this region. If ENC=00 bitfield CTXID is ignored. If BREN is set write to this bitfield is ignored.
00: If ENC=10 (resp. 11) the key stored in MCE_MKEYR (resp. MCE_FMKEYR) registers is used by the block cipher. If ENC=01 bypass mode is selected instead of stream cipher.
01: If ENC=10 or 11 the key stored in MCE_CC1KEYR is used by the block cipher.
If ENC=01 the key stored in MCE_CC1KEYR is used by the stream cipher. The nonce in MCE_CC1NRx registers and the version in MCE_CC1CR register are also used.
10: If ENC=10 or 11 the key stored in MCE_CC2KEYR is used by the block cipher.
If ENC=01 the key stored in MCE_CC2KEYR is used by the stream cipher. The nonce in MCE_CC2NRx registers and the version in MCE_CC2CR register are also used.
11: reserved

Bits 8:1  Reserved, must be kept at reset value.

Bit 0  BREN: Base region enable
0: Region x is disabled. Access control of primary region (privileged or unprivileged, no encryption) applies to any access between this region start and end addresses.
1: Region x is enable. Access controls and encryption option defined in this region apply to any access between this region start and end addresses.
BREN cannot be set if BADDRSTART > BADDREN.
39.6.10 MCE start address for region x register (MCE_SADDRx)

Address offset: 0x044 + 0x10 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR. Writes are ignored if GLOCK is set in MCE_CR register, or if BREN is set in MCE_REGCRx.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:12 BADDSTART[31:12]: Region address start
This bitfield defines the absolute start address of the region x on 4 kBytes boundary (inclusive).
BREN cannot be set if BADDRSTART > BADDREND.
When MCE determines the region, the first 12 bits (LSB) and the last 4 bits (MSB) in this register are ignored, and when this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros (reference value in MCE).

Bits 11:0 Reserved, must be kept at reset value.

39.6.11 MCE end address for region x register (MCE_EADDRx)

Address offset: 0x048 + 0x10 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0FFF

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR. Writes are ignored if GLOCK is set in MCE_CR register, or if BREN is set in MCE_REGCRx.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:12 BADDEND[31:12]: Region address end
This bitfield defines the absolute end address of the region x on 4 kBytes boundary (inclusive).
BREN cannot be set if BADDRSTART > BADDREND.
When MCE determines the region, the first 12 bits (LSB) and the last 4 bits (MSB) in this register are ignored, and when this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones (reference value in MCE).

Bits 11:0 Reserved, must be kept at reset value.
39.6.12 MCE attribute for region x register (MCE_ATTRx)

Address offset: 0x04C + 0x10 * (x - 1), (x = 1 to 4)
Reset value: 0x0000 0000
Privileged read and write access only if PRIV is set in MCE_PRIVCFGR. Writes are ignored if GLOCK is set in MCE_CR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WREN</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 16 **WREN**: Write enable
This bit is taken into account only if BREN is set.
0: Writes to region x are ignored. Reads are allowed.
1: Region x can be read and written. Restrictions linked to PRIV bit in MCE_REGCRx apply.

39.6.13 MCE master key x (MCE_MKEYRx)

Address offset: 0x200 + 0x4 * x, (x = 0 to 3)
Reset value: 0x0000 0000
Privileged write access only if PRIV is set in MCE_PRIVCFGR.
Writes are ignored if MKLOCK or GLOCK bit is set in MCE_CR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **MKEY{32 * x + i}**: Master key bit (i = 0 to 31)
This key is used by the MCE block cipher in normal, SCA resistant mode, if CTXID=0x0 in MCE_REGCR register.

39.6.14 MCE fast master key x (MCE_FMKEYRx)

Address offset: 0x220 + 0x4 * x, (x = 0 to 3)
Reset value: 0x0000 0000
Privileged write access only if PRIV is set in MCE_PRIVCFGR.
39.6.15 MCE cipher context z configuration register (MCE_CCzCFGR)

Address offset: 0x240 + 0x30 * (z - 1) (z = 1 to 2)
Reset value: 0x0000 0000

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR.

Writes are ignored if CCLOCK bit is set in this register. Writes are also ignored if GLOCK is set in MCE_CR register.

Refer to Section 39.3 to verify if this instance of MCE implements those registers.
Bits 31:16 **VERSION[15:0]:** Version

This 16-bit bitfield must be correctly initialized before CCEN bit is set. Bitfield usage is defined in Section 39.4.6: MCE stream cipher encryption mode.

Bits 15:8 **KEYCRC[7:0]:** Key CRC

When KEYLOCK=0, KEYCRC information is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new KEYCRC computation starts as soon as a new valid sequence is initiated. KEYCRC bitfield reads as zero until a valid sequence is completed (after it return the computed CRC value).

When GLOCK=1, KEYCRC bitfield always return the computed CRC value until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm \( X^8 + X^2 + X + 1 \) (according the convention).

*Note: CRC information is updated, and the key is usable by MCE, only after the last bit of the key has been written.*

When GLOCK=0 any write to MCE_CCxKEYR registers clears KEYCRC in MCE_CCxCFGR, and makes the cipher context key un-usable (bypass mode is selected instead). To be able to use the key again application must perform this sequence: write to KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). As KEYLOCK=1 all those writes are ignored, so the correct key is used instead.

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

*Bit 2** KEYLOCK: Key lock

0: Writes to MCE_CCzKEYR registers are allowed
1: Writes to MCE_CCzKEYR registers are ignored until next MCE reset.

*Note: This bit is set once. If this bit is set, it can only be cleared to 0 if MCE is reset.*

*Bit 1** CCLOCK: Cipher context lock

0: Writes to MCE_CCzCFGR and MCE_CCzNR registers are allowed
1: Writes to MCE_CCzCFGR and MCE_CCzNR registers are ignored until next MCE reset.

*Note: This bit is set once. If this bit is set, it can only be cleared to 0 if MCE is reset. Setting this bit forces KEYLOCK bit to 1.*

*Bit 0** CCEN: Cipher context enable

0: If an enabled region selects CTXID=z bypass mode is selected by MCE
1: If an enabled region selects CTXID=z with ENC=01 MCE selects with the stream cipher the information stored in MCE_CCzCFGR, MCE_CCzNR and MCE_CCzKEYR. If the enabled region selects ENC=10 or 11 instead MCE selects with the block cipher the key information stored in MCE_CCzKEYR.
39.6.16 MCE cipher context z nonce register 0 (MCE_CCzNR0)

Address offset: 0x244 + 0x30 *(z - 1) (z = 1 to 2)
Reset value: 0x0000 0000

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR.
Writes are ignored if CCLOCK is set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

Refer to Section 39.3 to verify if this instance of MCE implements those registers.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

SCNONCE[31:16]

Bits 31:0 SCNONCE[31:0]: Stream cipher nonce, bits [31:0]
This register is used by stream cipher to compute keystream. It must be correctly initialize before CCEN bit is set in MCE_CCzCFGR register. Bitfield usage is defined in Section 39.4.6: MCE stream cipher encryption mode.

39.6.17 MCE cipher context z nonce register 1 (MCE_CCzNR1)

Address offset: 0x248 + 0x30 *(z - 1) (z = 1 to 2)
Reset value: 0x0000 0000

Privileged read and write access only if PRIV is set in MCE_PRIVCFGR.
Writes are ignored if CCLOCK is set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

Refer to Section 39.3 to verify if this instance of MCE implements those registers.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

SCNONCE[63:48]

<table>
<thead>
<tr>
<th>63</th>
<th>62</th>
<th>61</th>
<th>60</th>
<th>59</th>
<th>58</th>
<th>57</th>
<th>56</th>
<th>55</th>
<th>54</th>
<th>53</th>
<th>52</th>
<th>51</th>
<th>50</th>
<th>49</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

SCNONCE[47:32]

Bits 31:0 SCNONCE[63:32]: Stream cipher nonce, bits [63:32]
Refer to the MCE_CCzNR0 register for description of the SCNONCE[63:0] bitfield.
39.6.18 MCE cipher context z key register 0 (MCE_CCzKEYR0)

Address offset: 0x24C + 0x30 *(z - 1) (z = 1 to 2)
Reset value: 0x0000 0000
Privileged write access only if PRIV is set in MCE_PRIVCFGR.
Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.
Refer to Section 39.3 to verify if this instance of MCE implements those registers.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

Bits 31:0 KEY[31:0]: cipher key, bits [31:0]
This register is used by the block or stream cipher of MCE when CTXID = z in encrypted region configuration register. KEY[127:0] must be correctly initialize before CCEN bit is set in MCE_CCzCFGR register.

39.6.19 MCE cipher context z key register 1 (MCE_CCzKEYR1)

Address offset: 0x250 + 0x30 *(z - 1) (z = 1 to 2)
Reset value: 0x0000 0000
Privileged write access only if PRIV is set in MCE_PRIVCFGR.
Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.
Refer to Section 39.3 to verify if this instance of MCE implements those registers.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
</tbody>
</table>

Bits 31:0 KEY[63:32]: cipher key, bits [63:32]
Refer to the MCE_CCzKEYR0 register for description of the KEY[127:0] bitfield.
### 39.6.20 MCE cipher context z key register 2 (MCE_CCzKEYR2)

Address offset: \(0x254 + 0x30 \times (z - 1)\) (\(z = 1\) to \(2\))  
Reset value: \(0x0000\ 0000\)  
Privileged write access only if PRIV is set in MCE_PRIVCFGR.  
Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.  
Refer to Section 39.3 to verify if this instance of MCE implements those registers.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>w</td>
</tr>
<tr>
<td>30</td>
<td>w</td>
</tr>
<tr>
<td>29</td>
<td>w</td>
</tr>
<tr>
<td>28</td>
<td>w</td>
</tr>
<tr>
<td>27</td>
<td>w</td>
</tr>
<tr>
<td>26</td>
<td>w</td>
</tr>
<tr>
<td>25</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>w</td>
</tr>
<tr>
<td>23</td>
<td>w</td>
</tr>
<tr>
<td>22</td>
<td>w</td>
</tr>
<tr>
<td>21</td>
<td>w</td>
</tr>
<tr>
<td>20</td>
<td>w</td>
</tr>
<tr>
<td>19</td>
<td>w</td>
</tr>
<tr>
<td>18</td>
<td>w</td>
</tr>
<tr>
<td>17</td>
<td>w</td>
</tr>
<tr>
<td>16</td>
<td>w</td>
</tr>
</tbody>
</table>

### 39.6.21 MCE cipher context z key register 3 (MCE_CCzKEYR3)

Address offset: \(0x258 + 0x30 \times (z - 1)\) (\(z = 1\) to \(2\))  
Reset value: \(0x0000\ 0000\)  
Privileged write access only if PRIV is set in MCE_PRIVCFGR.  
Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.  
Refer to Section 39.3 to verify if this instance of MCE implements those registers.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>w</td>
</tr>
<tr>
<td>30</td>
<td>w</td>
</tr>
<tr>
<td>29</td>
<td>w</td>
</tr>
<tr>
<td>28</td>
<td>w</td>
</tr>
<tr>
<td>27</td>
<td>w</td>
</tr>
<tr>
<td>26</td>
<td>w</td>
</tr>
<tr>
<td>25</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>w</td>
</tr>
<tr>
<td>23</td>
<td>w</td>
</tr>
<tr>
<td>22</td>
<td>w</td>
</tr>
<tr>
<td>21</td>
<td>w</td>
</tr>
<tr>
<td>20</td>
<td>w</td>
</tr>
<tr>
<td>19</td>
<td>w</td>
</tr>
<tr>
<td>18</td>
<td>w</td>
</tr>
<tr>
<td>17</td>
<td>w</td>
</tr>
<tr>
<td>16</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:0 **KEY[95:64]**: cipher key, bits [95:64]  
Refer to the MCE_CCzKEYR0 register for description of the KEY[127:0] bitfield.
## 39.6.22 MCE register map

| Offset | Register name      | Register name (reset value) | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------------|----------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | MCE_CR             | MCE_CR                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | MCE_SR             | MCE_SR                     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | MCE_IASR           | MCE_IASR                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | MCE_IACR           | MCE_IACR                   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | MCE_IAIER          | MCE_IAIER                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | MCE_PRIVCFGR       | MCE_PRIVCFGR               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | MCE_IAESR          | MCE_IAESR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | MCE_IADDR          | MCE_IADDR                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x040+ | MCE_REGCRx         | MCE_REGCRx                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x044+ | MCE_SADDRx         | MCE_SADDRx                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x048+ | MCE_EADDRx         | MCE_EADDRx                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04C+ | MCE_ATTRx          | MCE_ATTRx                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x200+ | MCE_MKEYRx         | MCE_MKEYRx                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x220+ | MCE_FMKEYRx        | MCE_FMKEYRx                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x240  | MCE_CC1CFGR        | MCE_CC1CFGR                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

### Table 362. MCE register map and reset values

- **Offset**: Memory address offset from the start of the register map.
- **Register name**: Name of the register.
- **Register name (reset value)**: Initial value of the register.
- **31**: Value at bit 31.
- **30**: Value at bit 30.
- **29**: Value at bit 29.
- **28**: Value at bit 28.
- **27**: Value at bit 27.
- **26**: Value at bit 26.
- **25**: Value at bit 25.
- **24**: Value at bit 24.
- **23**: Value at bit 23.
- **22**: Value at bit 22.
- **21**: Value at bit 21.
- **20**: Value at bit 20.
- **19**: Value at bit 19.
- **18**: Value at bit 18.
- **17**: Value at bit 17.
- **16**: Value at bit 16.
- **15**: Value at bit 15.
- **14**: Value at bit 14.
- **13**: Value at bit 13.
- **12**: Value at bit 12.
- **11**: Value at bit 11.
- **10**: Value at bit 10.
- **9**: Value at bit 9.
- **8**: Value at bit 8.
- **7**: Value at bit 7.
- **6**: Value at bit 6.
- **5**: Value at bit 5.
- **4**: Value at bit 4.
- **3**: Value at bit 3.
- **2**: Value at bit 2.
- **1**: Value at bit 1.
- **0**: Value at bit 0.

### Memory Address Calculations
- **Offset**: `0x000` to `0x240`.
- **Reset value**: `0x000` to `0x04C`.

### Register Descriptions
- **MCE_CR**: Memory cipher engine control register.
- **MCE_SR**: Memory cipher engine status register.
- **MCE_IASR**: Memory cipher engine input status register.
- **MCE_IACR**: Memory cipher engine input access control register.
- **MCE_IAIER**: Memory cipher engine input access control enable register.
- **MCE_PRIVCFGR**: Memory cipher engine private configuration register.
- **MCE_IAESR**: Memory cipher engine input access enable status register.
- **MCE_IADDR**: Memory cipher engine input address register.
- **MCE_REGCRx**: Memory cipher engine register control register `x`.
- **MCE_SADDRx**: Memory cipher engine source address register `x`.
- **MCE_EADDRx**: Memory cipher engine end address register `x`.
- **MCE_ATTRx**: Memory cipher engine attribute register `x`.
- **MCE_MKEYRx**: Memory cipher engine key register `x`.
- **MCE_FMKEYRx**: Memory cipher engine frequency key register `x`.
- **MCE_CC1CFGR**: Memory cipher engine configuration register.

### Reset Values
- **Reset value**: `0x000` to `0x04C`.

### Memory Address Calculations
- **0x000**: Start of the register map.
- **0x200**: Start of the key registers.
- **0x220**: Start of the frequency key registers.
- **0x240**: Start of the configuration register.

### Additional Information
- **Offset 0x040**: MCE_REGCRx register.
- **Offset 0x044**: MCE_SADDRx register.
- **Offset 0x048**: MCE_EADDRx register.
- **Offset 0x04C**: MCE_ATTRx register.
- **Offset 0x200**: MCE_MKEYRx register.
- **Offset 0x220**: MCE_FMKEYRx register.
- **Offset 0x240**: MCE_CC1CFGR register.

### STMicroelectronics
- **STMicroelectronics**: Manufacturer of the memory cipher engine (MCE).

### Revision Details
- **RM0477**: Revision of the document.
- **1594/3791**: Document identification number.
Table 362. MCE register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x244</td>
<td>MCE_CC1NR0</td>
<td>SCNONSE[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x248</td>
<td>MCE_CC1NR1</td>
<td>SCNONSE[63:32]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x24C</td>
<td>MCE_CC1KEYR0</td>
<td>KEY[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x250</td>
<td>MCE_CC1KEYR1</td>
<td>KEY[63:32]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x254</td>
<td>MCE_CC1KEYR2</td>
<td>KEY[95:64]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x258</td>
<td>MCE_CC1KEYR3</td>
<td>KEY[127:96]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x270</td>
<td>MCE_CC2CFGR</td>
<td>VERSION[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEYCRC[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>KEYLOCK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCLOCK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCEN</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x274</td>
<td>MCE_CC2NR0</td>
<td>SCNONSE[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x278</td>
<td>MCE_CC2NR1</td>
<td>SCNONSE[63:32]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x27C</td>
<td>MCE_CC2KEYR0</td>
<td>KEY[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x280</td>
<td>MCE_CC2KEYR1</td>
<td>KEY[63:32]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x284</td>
<td>MCE_CC2KEYR2</td>
<td>KEY[95:64]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x288</td>
<td>MCE_CC2KEYR3</td>
<td>KEY[127:96]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
40  Public key accelerator (PKA)

40.1  Introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, all needed computations are performed within the accelerator, so no further hardware/software elaboration is needed to process the inputs or the outputs.

When manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3.

40.2  PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), certified SESIP and PSA security assurance level 3
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
40.3 PKA functional description

40.3.1 PKA block diagram

Figure 396. PKA block diagram

40.3.2 PKA internal signals

Table 363 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pka_hclk</td>
<td>Digital input</td>
<td>AHB bus clock</td>
</tr>
<tr>
<td>pka_it</td>
<td>Digital output</td>
<td>Public key accelerator IP global interrupt request</td>
</tr>
<tr>
<td>pka_itamp_out</td>
<td>Digital output</td>
<td>PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.</td>
</tr>
</tbody>
</table>

40.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to Section 40.3.6 for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to Tamper detection and response in the System security section (if applicable to this product).
40.3.4  PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b \pmod{p}$. More information can be found in Section 40.5.1.

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

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in Table 364 and Table 365, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 364. PKA integer arithmetic functions list

<table>
<thead>
<tr>
<th>PKA_CR.MODE[5:0]</th>
<th>Performed operation</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>Montgomery parameter computation $R_2 \mod n$</td>
<td>Section 40.4.2</td>
</tr>
<tr>
<td>0x0E</td>
<td>Modular addition $(A+B) \mod n$</td>
<td>Section 40.4.3</td>
</tr>
<tr>
<td>0x0F</td>
<td>Modular subtraction $(A-B) \mod n$</td>
<td>Section 40.4.4</td>
</tr>
<tr>
<td>0x10</td>
<td>Montgomery multiplication $(A\times B) \mod n$</td>
<td>Section 40.4.5</td>
</tr>
<tr>
<td>0x00</td>
<td>Modular exponentiation $A^e \mod n$</td>
<td>Section 40.4.6</td>
</tr>
<tr>
<td>0x02</td>
<td>Modular exponentiation $A^e \mod n$ (fast mode)</td>
<td>Section 40.4.6</td>
</tr>
<tr>
<td>0x03</td>
<td>Modular exponentiation $A^e \mod n$ (protected)</td>
<td>Section 40.4.6</td>
</tr>
<tr>
<td>0x08</td>
<td>Modular inversion $A^{-1} \mod n$</td>
<td>Section 40.4.7</td>
</tr>
<tr>
<td>0x0D</td>
<td>Modular reduction $A \mod n$</td>
<td>Section 40.4.8</td>
</tr>
<tr>
<td>0x09</td>
<td>Arithmetic addition $A+B$</td>
<td>Section 40.4.9</td>
</tr>
<tr>
<td>0x0A</td>
<td>Arithmetic subtraction $A-B$</td>
<td>Section 40.4.10</td>
</tr>
<tr>
<td>0x0B</td>
<td>Arithmetic multiplication $A\times B$</td>
<td>Section 40.4.11</td>
</tr>
<tr>
<td>0x0C</td>
<td>Arithmetic comparison $(A=B, A&gt;B, A&lt;B)$</td>
<td>Section 40.4.12</td>
</tr>
<tr>
<td>0x07</td>
<td>RSA CRT exponentiation</td>
<td>Section 40.4.13</td>
</tr>
</tbody>
</table>
Each of these operating modes has an associated code that has to be written to the MODE field in the PKA_CR register. If the application selects any value that is not documented below the write to MODE bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in Table 364 and Table 365 are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reason the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming the application can decide to use a faster mode of operation, during which the precomputed Montgomery parameter is supplied before starting the operation. Performance improvement is detailed in Section 40.5.2: Computation times.

The only operation using fast mode is modular exponentiation (MODE = 0x02).
40.3.5 Typical applications for PKA

Introduction
The PKA can be used to accelerate a number of public key cryptographic functions. In particular:

• RSA encryption and decryption
• RSA key finalization
• CRT-RSA decryption
• DSA and ECDSA signature generation and verification
• DH and ECDH key agreement

Specifications of the above functions are given in following publications:

• FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
• PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories

The principles of the main functions are described in this section, for a more detailed description refer to the above cited documents.

RSA key pair
For the following RSA operations a public key and a private key information are defined as below:

• Alice transmits her public key \((n, e)\) to Bob. Numbers \(n\) and \(e\) are very large positive integers.
• Alice keeps secret her private key \(d\), also a very large positive integer. Alternatively this private key can also be represented by a quintuple \((p, q, dp, dq, q\text{inv})\).

For more information on the above representations refer to the RSA specification.

RSA encryption/decryption principle
As recommended by the PKCS#1 specification, Bob, to encrypt message \(M\) using Alice’s public key \((n, e)\) must go through the following steps:

1. Compute the encoded message \(EM = \text{ENCODE}(M)\), where ENCODE is an encoding method.
2. Turn \(EM\) into an integer \(m\), with \(0 \leq m < n\) and \((m, n)\) being coprimes.
3. Compute ciphertext \(c = m^e \mod n\).
4. Convert the integer \(c\) into a string ciphertext \(C\).
Alice, to decrypt ciphertext c using her private key d, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute \( \phi = (p - 1) \cdot (q - 1) \). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext \( m = c^d \mod n = (m^n)^d \mod n \). If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext m is obtained by performing the operations:
   a) \( m_1 = c^{dp} \mod p \)
   b) \( m_2 = c^{dq} \mod q \)
   c) \( h = qInv \cdot (m_1 - m_2) \mod p \)
   d) \( m = m_2 + h \cdot q \)
4. Convert the integer message representative m to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

Above operations can be accelerated by PKA using Modular exponentiation \( A^e \mod n \) if the private key is d, or RSA CRT exponentiation if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process protected version of modular exponentiation (MODE = 0x3) is strongly recommended for security reason. For encryption process MODE = 0x3 cannot be used, as it requires the knowledge of the private key.

Elliptic curve selection

For following ECC operations curve parameters are defined as below:
- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in Section 40.5.1: Supported elliptic curves.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, \( n \times G = \text{identity element} \ O \)).

ECDSA message signature generation

ECDSA (elliptic curve digital signature algorithm) signature generation function principle is the following: Alice, to sign a message m using her private key integer dA, goes through the following steps.

1. Calculate \( e = HASH(m) \), where HASH is a cryptographic hash function.
2. Let \( z \) be the \( L_n \) leftmost bits of e, where \( L_n \) is the bit length of the group order n.
3. Select a cryptographically secure random integer k where \( 0 < k < n \).
4. Calculate the curve point \((x_1, y_1) = k \times G\).
5. Calculate \( r = x_1 \mod n \). If \( r = 0 \) go back to step 3.
6. Calculate \( s = k^{-1} \cdot (z + rd_A) \mod n \). If \( s = 0 \) go back to step 3.
7. The signature is the pair \((r, s)\).
Steps 4 to 7 are accelerated by PKA using:

- **ECDSA sign** or
- All of the operations below:
  - **ECC Fp scalar multiplication** \( k \times P \)
  - **Modular reduction** \( A \mod n \)
  - **Modular inversion** \( A^{-1} \mod n \)
  - **Modular addition** and **Modular and Montgomery multiplication**

**ECDSA signature verification**

ECDSA (elliptic curve digital signature algorithm) signature verification function principle is the following: Bob, to authenticate Alice's signature, must have a copy of her public key curve point \( Q_A \).

Bob can verify that \( Q_A \) is a valid curve point going through the following steps:

1. check that \( Q_A \) is not equal to the identity element \( O \)
2. check that \( Q_A \) is on the agreed curve
3. check that \( n \times Q_A = O \).

Then Bob follows the procedure detailed below:

1. verify that \( r \) and \( s \) are integer in \([1, n-1]\)
2. calculate \( e = \text{HASH}(m) \), where \( \text{HASH} \) is the agreed cryptographic hash function
3. let \( z \) be the \( L_n \) leftmost bits of \( e \)
4. calculate \( w = s^{-1} \mod n \)
5. calculate \( u_1 = zw \mod n \) and \( u_2 = rw \mod n \)
6. calculate the curve point \((x_1, y_1) = u_1 \times G + u_2 \times Q_A\)
7. the signature is valid if \( r = x_1 \mod n \), it is invalid otherwise.

Steps 4 to 7 are accelerated by PKA using **ECDSA verification**.

### 40.3.6 PKA procedure to perform an operation

**Enabling/disabling PKA**

Setting the EN bit to 1 in PKA_CR register enables the PKA peripheral. The PKA becomes available when INITOK bit is set in PKA_SR. When EN = 0, the PKA peripheral is kept under reset, with PKA memory still accessible by the application through the AHB interface.

*Note: When PKA is in the process of clearing its memory EN bit cannot be set.*

When setting EN bit in PKA_CR make sure that the value of MODE bitfield corresponds to an authorized PKA operation (see OPERRF in Section 40.3.7).

Clearing EN bit to 0 while a calculation is in progress causes the operation to be aborted. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only PKA registers can be accessed, with writes to EN bits ignored.

If INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable PKA again.
Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in Section 40.4.

Executing a PKA operation

Each of the supported PKA operation is executed using the following procedure:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE field of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1) any access by the application to PKA RAM is ignored, and the flag RAMERRF is set in PKA_SR.

Selecting an illegal or unknown operation in step 2 triggers an OPERRF error, and step 3 (PROCENDF = 1) never happens. See Section 40.3.7 for details.

Using precomputed Montgomery parameters (PKA Fast mode)

As explained in Section 40.3.4, when computing many operations with the same modulus it can be beneficial for the application to compute only once the corresponding Montgomery parameter (see, for example, Section 40.4.5). This is know as “Fast mode”.

To manage the usage of Fast mode it is recommended to follow the procedure described below:

1. Load in PKA RAM the modulus size and value information. Such information is compiled in Section 40.5.1.
2. Program in PKA_CR register the PKA in Montgomery parameter computation mode (MODE="0x1") then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set to 1, then read back from PKA memory the corresponding Montgomery parameter, and then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.
4. Proceed with the required PKA operation, loading on top of regular input data the Montgomery information R2 mod m. All addresses are indicated in Section 40.4.

40.3.7 PKA error management

When PKA is used some errors can occur:

- The access to PKA RAM falls outside the expected range. In this case the address error flag (ADDRERRF) is set in the PKA_SR register.
- An AHB access to the PKA RAM occurred while the PKA core was using it. In this case the RAM error flag (RAMERRF) is set in the PKA_SR register, reads to PKA RAM return 0, while writes are ignored.
- The selected operating mode using MODE bitfield is not listed in PKA operating modes (see bitfield description), or PKA is running in limited mode (see LMF bit in PKA_SR). In this case the operation error flag (OPERRF) is set in the PKA_SR register, and write to MODE bitfield is ignored.
For each error flag above PKA generates an interrupt if the application sets the corresponding bit in PKA_CR register (see Section 40.6 for details).

ADDRERRF, OPERRF and RAMERRF errors are cleared by setting the corresponding bit in PKA_CLRFR.

OPERRF error must be cleared using OPERRFC bit in PKA_CLRFR before a new operation is written in PKA_CR register.

**Note:** The PKA can be re-initialized at any moment by resetting the EN bit in the PKA_CR register.

### 40.4 PKA operating modes

#### 40.4.1 Introduction

The various operations supported by PKA are described in the following subsections, defining the format of the input data and of the results, both stored in the PKA RAM.

---

**Warning:** The validity of all input parameters to the PKA must be checked before starting any operation, as PKA assumes that all of them are valid and consistent with each other. Input parameters must not exceed the operand size specified in the operation tables.

---

The following information applies to all PKA operations.

- PKA core processes 64-bit words in its RAM. Hence hereafter all word size is 64-bit.
- When an element is written as input in the PKA RAM, an additional word with all bits equal to zero has to be added after the most significant input word. This rule does not apply if the operand has a fixed size of 1.
- All reported RAM storage addresses refer to the least significant word of the data, and to obtain the actual address to use application must add to the indicated offset the base address of the PKA.
- Supported operand “Size” are:
  - ROS (RSA operand size): data size is \( \lceil \text{rsa\_size} / 64 \rceil + 1 \) words, with \( \text{rsa\_size} \) equal to the chosen modulus length in bits. For example, when computing RSA with an operand size of 1024 bits, ROS is equal to 17 words, or 1088 bits.
  - EOS (ECC operand size): data size is \( \lceil \text{ecc\_size} / 64 \rceil + 1 \) words, with \( \text{ecc\_size} \) equal to the chosen prime modulus length in bits. For example, when computing ECC with an operand size of 192 bits, EOS is equal to 4 words, or 256 bits.
  - ROS and EOS values include the required additional all 0 word.
- Unless indicated otherwise, all operands in the tables are integers.

**Note:** Fractional results for above formulas must be rounded up to the nearest integer since PKA core processes 64-bit words.

**Note:** The maximum ROS is 66 words (4160-bit max exponent size), while the maximum EOS is 11 words (640-bit max operand size).

As a first example (and to better understand the endianess in PKA memory), to prepare the operation ECC Fp scalar multiplication, when the application writes the x coordinate of point
P for an ECC P256 curve (EOS = 5 words), the least significant bit must be placed in bit 0 at address offset 0x578, and the most significant bit in bit 63 at address offset 0x590. Then, as mentioned above, the application must write the empty word 0x0000000000000000 at address offset 0x598.

As a second example, still to prepare the operation ECC Fp scalar multiplication, when the application need to write the information a = -3, on a curve with a modulus length of 224 bits (that is, four 64-bit words, rounded up, plus one) following data must be written in PKA memory:

@RAM+410 0x0000000000000001 /* curve coefficient 'a' sign without extra word */
@RAM+418 0x0000000000000011 /* value of |a| LSB */
@RAM+420 0x0000000000000000 ...
@RAM+428 0x0000000000000000 ...
@RAM+430 0x0000000000000000 value of |a| MSB */
@RAM+438 0x0000000000000000 /* additional all 0 word */

40.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter \((R^2 \mod n)\) used by PKA to convert operands into the Montgomery residue system representation. This operation can be very useful when fast mode operation is used, because in this case the Montgomery parameter is passed as input, saving the time for its computation.

Note: This operation can also be used with ECC curves. In this case prime modulus length and EOS size must be used.

Operation instructions for Montgomery parameter computation are summarized in
Table 366.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE</td>
<td>0x01</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(in bits, 0 ≤ value &lt; 4160)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Modulus value n</td>
<td>(odd integer only, n &lt; 2^{4160})</td>
<td>RAM@0x1088</td>
<td>ROS</td>
</tr>
</tbody>
</table>

40.4.3 Modular addition

Modular addition operation consists in the computation of \(A + B \mod n\). Operation instructions are summarized in Table 367.
40.4.4 Modular subtraction

Modular subtraction operation consists in the following computations:
- If \( A \geq B \) result equals \( A - B \mod n \)
- If \( A < B \) result equals \( A + n - B \mod n \)

Operation instructions are summarized in Table 368.

Table 368. Modular subtraction

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0F</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>((0 \leq A &lt; n))</td>
<td>RAM@0xA50</td>
<td></td>
</tr>
<tr>
<td>Operand B</td>
<td>((0 \leq B &lt; n))</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>Modulus value n</td>
<td>((n &lt; 2^{4160}))</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: A-B mod n ((0 \leq \text{result} &lt; n))</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
</tbody>
</table>

40.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications the PKA accelerates multiplication which has at least one input in the Montgomery domain. The two main uses of this operation are:
- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication \( A \times B \mod n \)

The method to perform above operations are described below. Note that “\( x \)” function is this operation, and \( A, B, C \) operands are in the natural domain.
1. Inward (or outward) conversion into (or from) Montgomery domain
   a) Assuming that A is an integer in the natural domain:
      – Compute \( r_2 \text{mod} n \) using Montgomery parameter computation.
      – Result \( AR = A \times r_2 \text{mod} n \) is A in the Montgomery domain.
   b) Assuming that \( BR \) is an integer in the Montgomery domain:
      – Result \( B = BR \times 1 \text{mod } n \) is B in the natural domain.
      – Similarly, above value \( AR \) computed in a) can be converted into the natural domain by computing \( A = AR \times 1 \text{mod } n \).

2. Simple modular multiplication \( A \times B \text{mod } n \)
   a) Compute \( r_2 \text{mod} n \) using Montgomery parameter computation.
   b) Compute \( AR = A \times r_2 \text{mod} n \). Output is in the Montgomery domain.
   c) Compute \( AB = AR \times B \text{mod } n \). Output is in natural domain.

3. Multiple modular multiplication \( A \times B \times C \text{mod } n \)
   a) Compute \( r_2 \text{mod} n \) using Montgomery parameter computation.
   b) Compute \( AR = A \times r_2 \text{mod} n \). Output is in the Montgomery domain.
   c) Compute \( BR = B \times r_2 \text{mod} n \). Output is in the Montgomery domain.
   d) Compute \( ABR = AR \times BR \text{mod } n \). Output is in the Montgomery domain.
   e) Compute \( CR = C \times r_2 \text{mod} n \). Output is in the Montgomery domain.
   f) Compute \( ABCR = ABR \times CR \text{mod } n \). Output is in the Montgomery domain.
   g) (optional) Repeat the two steps above if more operands need to be multiplied.
   h) Compute \( ABC = ABCR \times 1 \text{mod } n \) to retrieve the result in natural domain.

Operation instructions for Montgomery multiplication are summarized in Table 369.

### Table 369. Montgomery multiplication

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td>MODE 0x10</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td></td>
<td>Operand length (in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td></td>
<td>Operand A ((0 \leq A &lt; n))</td>
<td>RAM@0xA50</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Operand B ((0 \leq B &lt; n))</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modulus value n ((\text{odd integer only, } n &lt; 2^{4160}))</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: ( A \times B \text{mod } n )(^{(1)})</td>
<td>-</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples 2 and 3).

### 40.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It consists in the computation of \( A^e \text{mod } n \).

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on Table 370 and Table 371. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on Table 372, for security reason.

**Note:** Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

---

RM0477 Rev 6 1607/3791
When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in Table 370 (normal mode), Table 371 (fast mode) and in Table 372 (protected mode). Fast mode usage is explained in Section 40.3.6.

### Table 370. Modular exponentiation (normal mode)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE</td>
<td>0x00</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN Exponent length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
<td>64 bits</td>
</tr>
<tr>
<td>IN Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>IN/OUT Operand A (base of exponentiation)</td>
<td>$(0 \leq A &lt; n)$</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>IN Exponent e</td>
<td>$(0 \leq e &lt; n)$</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
<tr>
<td>IN Modulus value n</td>
<td>(odd integer only, $n &lt; 2^{4160}$)</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>OUT Result: $A^e \mod n$</td>
<td>$(0 \leq \text{result} &lt; n)$</td>
<td>RAM@0x838</td>
<td></td>
</tr>
</tbody>
</table>

### Table 371. Modular exponentiation (fast mode)

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE</td>
<td>0x02</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>IN Exponent length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
<td>64 bits</td>
</tr>
<tr>
<td>IN Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>IN/OUT Operand A (base of exponentiation)</td>
<td>$(0 \leq A &lt; n)$</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>IN Exponent e</td>
<td>$(0 \leq e &lt; n)$</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
<tr>
<td>IN Modulus value n</td>
<td>(odd integer only, $n &lt; 2^{4160}$)</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>IN/OUT Montgomery parameter R2 mod n</td>
<td>(mandatory)</td>
<td>RAM@0x620</td>
<td></td>
</tr>
<tr>
<td>OUT Result: $A^e \mod n$</td>
<td>$(0 \leq \text{result} &lt; n)$</td>
<td>RAM@0x838</td>
<td></td>
</tr>
</tbody>
</table>
40.4.7 Modular inversion

Modular inversion operation consists in the computation of multiplicative inverse \( A^{-1} \mod n \).

If the modulus \( n \) is prime, for all values of \( A (1 \leq A < n) \) modular inversion output is valid. If the modulus \( n \) is not prime, \( A \) has an inverse only if the greatest common divisor between \( A \) and \( n \) is 1.

If the operand \( A \) is a divisor of the modulus \( n \) the result is a multiple of a factor of \( n \).

Operation instructions for modular inversion are summarized in Table 373.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0x08</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>(0 \leq A &lt; n)</td>
<td>RAM@0xA50</td>
<td></td>
</tr>
<tr>
<td>Modulus value n</td>
<td>(odd integer only, n &lt; 2^{4160})</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: A^{-1} mod n</td>
<td>0 &lt; result &lt; n</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

40.4.8 Modular reduction

Modular reduction operation consists in the computation of the remainder of \( A \) divided by \( n \).

Operation instructions are summarized in Table 374.

---

1. Euler totient function of \( n \) with \( \phi = (p - 1) \times (q - 1) \), where \( p \) and \( q \) are prime factors of modulus \( n \) (see NIST SP800-56B Appendix C or RSA encryption/decryption principle for details). As optimization it is recommended to keep \( \phi \) information as part of the key pair generation. Alternative is to store the key as \((p, q, e, d)\) instead of \((N, e, d, \phi)\), in this case, to derive \( N \) and \( \phi \) using PKA arithmetic multiplier: \( N = p \times q \), and \( \phi = (p - 1) \times (q - 1) \).
40.4.9 Arithmetic addition

Arithmetic addition operation consists in the computation of $A + B$. Operation instructions are summarized in Table 375.

**Table 375. Arithmetic addition**

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x09</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>$0 \leq A &lt; 2^M$</td>
<td>RAM@0xA50</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>$0 \leq B &lt; 2^M$</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: A+B</td>
<td>RAM@0xE78</td>
<td>ROS + 1</td>
</tr>
</tbody>
</table>

40.4.10 Arithmetic subtraction

Arithmetic subtraction operation consists in the following computations:
- If $A \geq B$ result equals $A - B$
- If $A < B$ and $M/32$ residue is > 0 result equals $A + 2^{\text{int}(M/32)^*32+1} - B$
- If $A < B$ and $M/32$ residue is 0 result equals $A + 2^{\text{int}(M/32)^*32} - B$

For the last two bullets the 32-bit word following the most significant word of the output equals 0xFFFF FFFF, as result is negative.

Operation instructions are summarized in Table 376.

**Table 376. Arithmetic subtraction**

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0A</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>$0 \leq A &lt; 2^M$</td>
<td>RAM@0xA50</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>$0 \leq B &lt; 2^M$</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: A-B</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
</tbody>
</table>
40.4.11 Arithmetic multiplication

Arithmetic multiplication operation consists in the computation of AxB. Operation instructions are summarized in Table 377.

Table 377. Arithmetic multiplication

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0B</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xA50</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result: AxB</td>
<td>(0 ≤ result &lt; 2^{2M})</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

40.4.12 Arithmetic comparison

Arithmetic comparison operation consists in the following computation:
- If A = B then result = 0xED2C
- If A > B then result = 0x7AF8
- If A < B then result = 0x916A

Operation instructions for arithmetic comparison are summarized in Table 378.

Table 378. Arithmetic comparison

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x0C</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Operand length M</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Operand A</td>
<td>(0 ≤ A &lt; 2^M)</td>
<td>RAM@0xA50</td>
<td>ROS</td>
</tr>
<tr>
<td>Operand B</td>
<td>(0 ≤ B &lt; 2^M)</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td>Result A?B</td>
<td>0xED2C, 0x7AF8 or 0x916A</td>
<td>RAM@0xE78</td>
</tr>
</tbody>
</table>

40.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):
- p and q are precomputed primes, stored as part of the private key
- dp = d mod (p -1)
- dq = d mod (q -1) and
- dq = d mod (q -1) and
- d_{inv} = q^{-1} mod p
These values allow the recipient to compute the exponentiation $m = A^d \mod pq$ more efficiently as follows:

- $m_1 = A^{dP} \mod p$
- $m_2 = A^{dQ} \mod q$
- $h = q_{inv} (m_1 - m_2) \mod p$, with $m_1 > m_2$
- $m = m_2 + hq \mod pq$

Operation instructions for computing CRT exponentiation $A^d \mod pq$ are summarized in **Table 379**.

### Table 379. CRT exponentiation

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN MODE 0x07</td>
<td>6 bits</td>
<td>PKA_CR</td>
<td></td>
</tr>
<tr>
<td>IN Operand length</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>IN Operand $d_p$</td>
<td>$0 &lt; d_p &lt; 2^{M/2}$</td>
<td>RAM@0x730</td>
<td></td>
</tr>
<tr>
<td>IN Operand $d_Q$</td>
<td>$0 &lt; d_Q &lt; 2^{M/2}$</td>
<td>RAM@0xE78</td>
<td></td>
</tr>
<tr>
<td>IN Operand $q_{inv}$</td>
<td>$0 &lt; q_{inv} &lt; 2^{M/2}$</td>
<td>RAM@0x948</td>
<td></td>
</tr>
<tr>
<td>IN Prime $p$</td>
<td>$0 &lt; p &lt; 2^{M/2}$</td>
<td>RAM@0xB60</td>
<td></td>
</tr>
<tr>
<td>IN Prime $q$</td>
<td>$0 &lt; q &lt; 2^{M/2}$</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
<tr>
<td>IN Operand A</td>
<td>$0 &lt; A &lt; 2^M$</td>
<td>RAM@0x12A0</td>
<td></td>
</tr>
<tr>
<td>OUT Result: $A^d \mod pq$</td>
<td>($0 \leq \text{result} &lt; pq$)</td>
<td>RAM@0x838</td>
<td></td>
</tr>
</tbody>
</table>

1. Must be different from 2.

#### 40.4.14 Point on elliptic curve Fp check

This operation consists in checking whether a given point $P (x, y)$ satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \mod p$, where $a$ and $b$ are elements of the curve.

Operation instructions for point on elliptic curve Fp check are summarized in **Table 380**.
40.4.15 ECC Fp scalar multiplication

This operation consists in the computation of \( k \times P (x_P, y_P) \), where \( P \) is a point on a curve over prime fields and \( "x" \) is the elliptic curve scalar point multiplication. Result of the computation is a point that belongs to the same curve or a point at infinity.

Operation instructions for ECC Fp scalar multiplication are summarized in Table 381.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.

Table 381. ECC Fp scalar multiplication

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x20</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order ( n ) length</td>
<td>(in bits, not null,)</td>
<td>RAM@0x400</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve modulus ( p ) length</td>
<td>(in bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient ( a ) sign</td>
<td>0x0: positive 0x1: negative</td>
<td>RAM@0x410</td>
<td></td>
</tr>
</tbody>
</table>

Table 380. Point on elliptic curve Fp check

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x28</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Modulus length</td>
<td>(in bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient ( a ) sign</td>
<td>0x0: positive 0x1: negative</td>
<td>RAM@0x410</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient ( a )</td>
<td>(absolute value, ([a] &lt; p))</td>
<td>RAM@0x418</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient ( b )</td>
<td>([b] &lt; ( p ))</td>
<td>RAM@0x520</td>
<td></td>
</tr>
<tr>
<td>Curve modulus value ( p )</td>
<td>(odd integer prime, ( 0 &lt; p &lt; 2640 ))</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate ( x )</td>
<td>((x &lt; p))</td>
<td>RAM@0x578</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate ( y )</td>
<td>((y &lt; p))</td>
<td>RAM@0x5D0</td>
<td></td>
</tr>
<tr>
<td>Montgomery parameter ( R2 \mod n )</td>
<td>-</td>
<td>RAM@0x4C8</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Result: point ( P ) on curve</td>
<td>0xD60D: point on curve 0xA3B7: point not on curve 0xF946: ( x ) or ( y ) coordinate is not smaller than modulus ( p )</td>
<td>RAM@0x680</td>
<td>64 bits</td>
</tr>
</tbody>
</table>
When performing this operation the following special cases must be noted:

- For $k = 0$ this function returns a point at infinity $(0, 0)$ if curve parameter $b$ is nonzero, $(0, 1)$ otherwise. For $k$ different from 0 it might happen that a point at infinity is returned. When the application detects this behavior a new computation must be carried out.

- For $k < 0$ (that is, a negative scalar multiplication is required) the multiplier absolute value $k = |-k|$ must be provided to the PKA. After the computation completion, the formula $-P = (x, -y)$ can be used to compute the $y$ coordinate of the effective final result (the $x$ coordinate remains the same).

**Note:** The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

### 40.4.16 ECDSA sign

ECDSA signing operation (outlined in Section 40.3.5) is summarized in Table 382 (input parameters) and in Table 383 (output parameters).

The application has to check if the output error is equal to 0xD60D, if it is different a new $k$ must be generated and the ECDSA sign operation must be repeated.

**Note:** Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation "Point on elliptic curve" can be used to avoid this.
Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Extended ECDSA support

PKA also supports extended ECDSA signature, for which the inputs and the outputs are the same as ECDSA signature (Table 382 and Table 383, respectively), with the addition of the coordinates of the point kG. This extra output is defined in Table 384.
40.4.17 ECDSA verification

ECDSA verification operation (outlined in Section 40.3.5) is summarized in Table 385 (input parameters) and Table 386 (output parameters).

The application has to check if the output error is equal to 0xD60D, if different the signature is not verified.

### Table 384. Extended ECDSA sign - Extra outputs

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curve point kG coordinate x₁</td>
<td>(0 ≤ x₁ &lt; p)</td>
<td>RAM@0x1400</td>
<td>EOS</td>
</tr>
<tr>
<td>Curve point kG coordinate y₁</td>
<td>(0 ≤ y₁ &lt; p)</td>
<td>RAM@0x1458</td>
<td></td>
</tr>
</tbody>
</table>

### Table 385. ECDSA verification - Inputs

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x26</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order n length (nlen)</td>
<td>(in bits, not null)</td>
<td>RAM@0x408</td>
<td>EOS</td>
</tr>
<tr>
<td>Curve modulus p length</td>
<td>(in bits, not null, 8 &lt; value &lt; 640)</td>
<td>RAM@0x4C8</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve coefficient a sign</td>
<td>0x0: positive</td>
<td>RAM@0x468</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x1: negative</td>
<td>RAM@0x468</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient</td>
<td>(absolute value,</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Curve modulus value p</td>
<td>(odd integer prime, 0 &lt; p &lt; 2&lt;sup&gt;640&lt;/sup&gt;)</td>
<td>RAM@0x4D0</td>
<td></td>
</tr>
<tr>
<td>Curve base point G coordinate x</td>
<td>(x &lt; p)</td>
<td>RAM@0x678</td>
<td></td>
</tr>
<tr>
<td>Curve base point G coordinate y</td>
<td>(y &lt; p)</td>
<td>RAM@0x6D0</td>
<td>EOS</td>
</tr>
<tr>
<td>Public-key curve point Q coordinate xₚ</td>
<td>(xₚ &lt; p)</td>
<td>RAM@0x12F8</td>
<td></td>
</tr>
<tr>
<td>Public-key curve point Q coordinate yₚ</td>
<td>(yₚ &lt; p)</td>
<td>RAM@0x1350</td>
<td></td>
</tr>
<tr>
<td>Signature part r</td>
<td>(0 &lt; r &lt; n)</td>
<td>RAM@0x10E0</td>
<td></td>
</tr>
<tr>
<td>Signature part s</td>
<td>(0 &lt; s &lt; n)</td>
<td>RAM@0xC68</td>
<td></td>
</tr>
<tr>
<td>Hash of message z</td>
<td>(hash size equal to nlen)&lt;sup&gt;(1)&lt;/sup&gt;</td>
<td>RAM@0x13A8</td>
<td></td>
</tr>
<tr>
<td>Curve prime order n</td>
<td>(integer prime)</td>
<td>RAM@0x1088</td>
<td></td>
</tr>
</tbody>
</table>

1. Padding with zeroes or hash truncation must be used to have the hash parameter size equal to the curve prime order n length.
40.4.18 ECC complete addition

ECC complete addition computes the addition of two given points on an elliptic curve.

Operation instructions are summarized in Table 387.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x,y) conversion (X,Y,Z) = (x,y,1) can be used. To convert resulting point to Jacobian coordinates conversion (x,y) = (X/Z², Y/Z³) can be used.

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUT Result: ECDSA verify</td>
<td>– 0xD60D: valid signature – 0xA3B7: invalid signature</td>
<td>RAM@0x5D0</td>
<td>64 bits</td>
</tr>
<tr>
<td></td>
<td>Computed signature part r</td>
<td>– (0 &lt; r &lt; n)</td>
<td>RAM@0x578</td>
</tr>
</tbody>
</table>

40.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of k*P+m*Q, where (P,Q) are two points on an elliptic curve and (k,m) are two scalars. Operation instructions are summarized in Table 388.

If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0, 0).
Note: The two input points are represented in Jacobian coordinates \((X, Y, Z)\). To input a point in affine coordinates \((x, y)\) conversion \((X, Y, Z) = (x, y, 1)\) can be used. The result is represented in affine coordinates \((x, y)\).

This operation requires the input point \(Z\) coordinate to be equal to 1 (input point represented in affine coordinates).

### Table 388. ECC double base ladder

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0x27</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve prime order (n) length</td>
<td>(in bits, not null)</td>
<td>RAM@0x400</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve modulus (p) length</td>
<td>(in bits, not null, (8 &lt; \text{value} &lt; 640))</td>
<td>RAM@0x408</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient (a) sign</td>
<td>− 0x0: positive − 0x1: negative</td>
<td>RAM@0x410</td>
<td></td>
</tr>
<tr>
<td>Curve coefficient (</td>
<td>a</td>
<td>)</td>
<td>(absolute value, (</td>
</tr>
<tr>
<td>Curve modulus value (p)</td>
<td>(odd integer prime, (0 &lt; p &lt; 2^{640}))</td>
<td>RAM@0x470</td>
<td></td>
</tr>
<tr>
<td>Integer (k)</td>
<td>((0 &lt; k &lt; 2^{640}))</td>
<td>RAM@0x520</td>
<td></td>
</tr>
<tr>
<td>Integer (m)</td>
<td>((0 &lt; m &lt; 2^{640}))</td>
<td>RAM@0x578</td>
<td></td>
</tr>
<tr>
<td>First point (P) coordinate (X)</td>
<td>((x &lt; p))</td>
<td>RAM@0x628</td>
<td></td>
</tr>
<tr>
<td>First point (P) coordinate (Y)</td>
<td>((y &lt; p))</td>
<td>RAM@0x680</td>
<td></td>
</tr>
<tr>
<td>First point (P) coordinate (Z)</td>
<td>((z &lt; p))</td>
<td>RAM@0x6D8</td>
<td></td>
</tr>
<tr>
<td>Second point (Q) coordinate (X)</td>
<td>((x &lt; p))</td>
<td>RAM@0x730</td>
<td></td>
</tr>
<tr>
<td>Second point (Q) coordinate (Y)</td>
<td>((y &lt; p))</td>
<td>RAM@0x788</td>
<td></td>
</tr>
<tr>
<td>Second point (Q) coordinate (Z)</td>
<td>((z &lt; p))</td>
<td>RAM@0x7E0</td>
<td></td>
</tr>
<tr>
<td>OUT Result coordinate (x)</td>
<td>((x &lt; p))</td>
<td>RAM@0x578</td>
<td>EOS</td>
</tr>
<tr>
<td>OUT Result coordinate (y)</td>
<td>((y &lt; p))</td>
<td>RAM@0x5D0</td>
<td></td>
</tr>
<tr>
<td>Error code</td>
<td>– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7</td>
<td>RAM@0x520</td>
<td>64 bits</td>
</tr>
</tbody>
</table>

### 40.4.20 ECC projective to affine

ECC projective to affine operation computes the conversion between the representation of a point \(P\) in homogeneous projective coordinates and the representation of the point \(P\) in affine coordinates. Namely, if the point is represented by the triple \((X, Y, Z)\), it computes the affine coordinates \((x, y) = (X/Z, Y/Z)\).

All the operations are performed modulo the modulus \(p\) of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals \((0,0)\).

Operation instructions are summarized in Table 389.
### 40.5 Example of configurations and processing times

#### 40.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b \pmod p$.

**Note:** Binary curves, Edwards curves and Curve25519 are not supported by the PKA. The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer $p$, used as the modulus for all point arithmetic in the finite field $GF(p)$
- the (usually prime) integer $n$, the order of the group generated by $G$, defined below
- the base point of the curve $G$, defined by its coordinates $(Gx, Gy)$
- the integers $a$ and $b$, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define $a$ as negative, PKA supports two representations:

1. **a defined as $p$-[a]** in the finite field $GF(p)$, for example $p$-3:
   - Curve coefficient $p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$  
   - Curve coefficient a sign= 0x0 (positive)
   - Curve coefficient a = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF

2. **a defined as negative**, for example -3:
   - Curve coefficient $p = 0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF$  
   - Curve coefficient a sign= 0x1 (negative)
   - Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000003

---

### Table 389. ECC projective to affine

<table>
<thead>
<tr>
<th>Parameters with direction</th>
<th>Value (note)</th>
<th>Storage</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>IN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MODE</td>
<td>0x2F</td>
<td>PKA_CR</td>
<td>6 bits</td>
</tr>
<tr>
<td>Curve modulus $p$ length</td>
<td>(in bits, $8 &lt; value &lt; 640$)</td>
<td>RAM@0x408</td>
<td>64 bits</td>
</tr>
<tr>
<td>Curve modulus value $p$</td>
<td>(odd integer prime, $0 &lt; p &lt; 2^{640}$)</td>
<td>RAM@0x470</td>
<td>64 bits</td>
</tr>
<tr>
<td>Point P coordinate X (projective)</td>
<td>$(x &lt; p)$</td>
<td>RAM@0xD60</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate Y (projective)</td>
<td>$(y &lt; p)$</td>
<td>RAM@0xDB8</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate Z (projective)</td>
<td>$(z &lt; p)$</td>
<td>RAM@0xE10</td>
<td></td>
</tr>
<tr>
<td>Montgomery parameter R2 mod n</td>
<td>-</td>
<td>RAM@0x4C8</td>
<td></td>
</tr>
<tr>
<td>OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Point P coordinate x (affine)</td>
<td>$(x &lt; p)$</td>
<td>RAM@0x578</td>
<td></td>
</tr>
<tr>
<td>Point P coordinate y (affine)</td>
<td>$(y &lt; p)$</td>
<td>RAM@0x5D0</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Error code</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Notes:*
- Point not at infinity: 0xD60D
- Point at infinity: 0xA3B7

*RAM@0x680 64 bits*
Table 390 summarizes the family of curves supported by PKA for ECC operations.

<table>
<thead>
<tr>
<th>Curve name</th>
<th>Standard</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-192</td>
<td>NIST</td>
<td>Digital Signature Standard (DSS), NIST FIPS 186-4</td>
</tr>
<tr>
<td>P-224</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P-521</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP224r1, brainpoolP224t1</td>
<td>IETF</td>
<td>– Brainpool Elliptic Curves, IETF RFC 5639</td>
</tr>
<tr>
<td>brainpoolP256r1, brainpoolP256t1</td>
<td></td>
<td>– Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry, IETF RFC 6932</td>
</tr>
<tr>
<td>brainpoolP320r1, brainpoolP320t1</td>
<td></td>
<td><a href="https://tools.ietf.org">https://tools.ietf.org</a></td>
</tr>
<tr>
<td>brainpoolP384r1, brainpoolP384t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>brainpoolP512r1, brainpoolP512t1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp192k1, secp192r1</td>
<td>SEC</td>
<td>Standards for Efficient Cryptography SEC 2 curves</td>
</tr>
<tr>
<td>secp224k1, secp224r1</td>
<td></td>
<td><a href="https://www.secg.org">https://www.secg.org</a></td>
</tr>
<tr>
<td>secp256k1, secp256r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp384r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>secp521r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recommended curve parameters for public key cryptographic algorithm SM2</td>
<td>OSCCA</td>
<td>– Public key cryptographic algorithm SM2 based on elliptic curves, Organization of State Commercial Administration of China OSCCA SM2, December 2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Digital signatures - Part 3 Discrete logarithm based mechanisms, ISO/IEC 14888-3, November 2018</td>
</tr>
</tbody>
</table>
40.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

### Table 391. Modular exponentiation

<table>
<thead>
<tr>
<th>Exponent length (in bits)</th>
<th>Mode</th>
<th>Modulus length (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Normal</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Normal</td>
<td>124600</td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>22700</td>
</tr>
<tr>
<td>17</td>
<td>Normal</td>
<td>135700</td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>33800</td>
</tr>
<tr>
<td>$2^{16} + 1$</td>
<td>Normal</td>
<td>180000</td>
</tr>
<tr>
<td></td>
<td>Fast</td>
<td>78200</td>
</tr>
</tbody>
</table>

#### Table 392. ECC scalar multiplication

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
</tr>
<tr>
<td>1590000</td>
</tr>
</tbody>
</table>

1. CRT stands for Chinese remainder theorem optimization (MODE bitfield= 0x07).

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.
Table 393. ECDSA signature average computation time

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1500</td>
<td>2744</td>
<td>4579</td>
<td>7184</td>
<td>1445</td>
<td>1668</td>
<td>2496</td>
<td>2958</td>
</tr>
</tbody>
</table>

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.

2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 394. ECDSA verification average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1011</td>
<td>1495</td>
<td>2938</td>
<td>5014</td>
<td>7979</td>
<td>1680</td>
<td>1925</td>
<td>2958</td>
</tr>
</tbody>
</table>

Table 395. ECC double base ladder average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>967</td>
<td>1419</td>
<td>2768</td>
<td>4784</td>
<td>7547</td>
<td>1585</td>
<td>1825</td>
<td>2825</td>
</tr>
</tbody>
</table>

Table 396. ECC projective to affine average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>476</td>
<td>7800</td>
<td>1483</td>
<td>2530</td>
<td>4190</td>
<td>8384</td>
<td>1049</td>
</tr>
</tbody>
</table>

Table 397. ECC complete addition average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1000</td>
<td>1200</td>
<td>1800</td>
<td>2600</td>
<td>3900</td>
<td>5300</td>
<td>8900</td>
</tr>
</tbody>
</table>

Table 398. Point on elliptic curve Fp check average computation times

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>160</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>384</th>
<th>512</th>
<th>521</th>
<th>640</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3400</td>
<td>4200</td>
<td>6100</td>
<td>8300</td>
<td>10900</td>
<td>17200</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see Section 40.3.7
2. Access to unmapped address (ADDRERRF), see Section 40.3.7
3. PKA RAM access while PKA operation is in progress (RAMERRF), see Section 40.3.7
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it. The user can enable or disable above interrupt sources individually by changing the mask bits in the PKA control register (PKA_CR). Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 400 gives a summary of the available features.

### Table 399. Montgomery parameters average computation times(1)

<table>
<thead>
<tr>
<th>Modulus length (in bits)</th>
<th>192</th>
<th>256</th>
<th>320</th>
<th>512</th>
<th>1024</th>
<th>2048</th>
<th>3072</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8600</td>
<td>8710</td>
<td>11870</td>
<td>17000</td>
<td>102000</td>
<td>410000</td>
<td>506000</td>
<td>822000</td>
</tr>
</tbody>
</table>

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

#### 40.6 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see Section 40.3.7
2. Access to unmapped address (ADDRERRF), see Section 40.3.7
3. PKA RAM access while PKA operation is in progress (RAMERRF), see Section 40.3.7
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it. The user can enable or disable above interrupt sources individually by changing the mask bits in the PKA control register (PKA_CR). Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 400 gives a summary of the available features.

### Table 400. PKA interrupt requests

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Clear method</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKA</td>
<td>Unsupported operation</td>
<td>OPERRF</td>
<td>OPERRIE</td>
<td>Set OPERRFC bit</td>
</tr>
<tr>
<td></td>
<td>Access to unmapped address</td>
<td>ADDRERRF</td>
<td>ADDRERRIE</td>
<td>Set ADDRERRFC bit</td>
</tr>
<tr>
<td></td>
<td>error</td>
<td>RAMERRF</td>
<td>RAMERRIE</td>
<td>Set RAMERRFC bit</td>
</tr>
<tr>
<td></td>
<td>PKA end of operation</td>
<td>PROCENDF</td>
<td>PROCENDIE</td>
<td>Set PROCENDFC bit</td>
</tr>
</tbody>
</table>
40.7   PKA registers

40.7.1   PKA control register (PKA_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:22</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

Bit 21   **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20   **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19   **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18   Reserved, must be kept at reset value.

Bit 17   **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14   Reserved, must be kept at reset value.
Bits 13:8 **MODE[5:0]**: PKA operation code

- 000000: Montgomery parameter computation then modular exponentiation
- 000001: Montgomery parameter computation only
- 000010: Modular exponentiation (protected, used when manipulating secrets)
- 100000: Montgomery parameter computation then ECC scalar multiplication (protected)
- 000100: ECDSA sign (protected)
- 100100: ECDSA verification
- 100110: ECC projective to affine
- 000011: Modular parameter computation only
- 000100: Modular exponentiation only (Montgomery parameter must be loaded first)
- 000110: Modular exponentiation
- 101000: Point on elliptic curve Fp check
- 000111: RSA CRT exponentiation
- 001000: Modular inversion
- 001001: Arithmetic addition
- 001010: Arithmetic subtraction
- 001011: Arithmetic multiplication
- 001100: Arithmetic comparison
- 001101: Modular reduction
- 001110: Modular addition
- 001111: Modular subtraction
- 001111: Modular subtraction
- 100000: Montgomery multiplication
- 100011: ECC complete addition
- 100111: ECC double base ladder
- 101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE bit is ignored.

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

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

**Note**: **START** is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

- 0: Disable PKA
- 1: Enable PKA. PKA becomes functional when INITOK is set by hardware in PKA_SR.

When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE bitfield for details.

**Note**: When EN = 0, PKA RAM can still be accessed by the application.
### 40.7.2 PKA status register (PKA_SR)

Address offset: 0x04  
Reset value: 0x0000 0000

| Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserved| Reserve| OPERRF | ADDRERRF| RAMERRF| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| Reserve| LMF | INITOK |
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      | 0      |

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

Bit 21 **OPERRF**: Operation error flag
- 0: No event error
- 1: An illegal or unknown operation has been selected in PKA_CR register
This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag
- 0: No address error
- 1: Address access is out of range (unmapped address)
This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag
- 0: No PKA RAM access error
- 1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).
This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag
- 0: Operation in progress
- 1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag
This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.
- 0: No operation is in progress (default)
- 1: An operation is in progress
If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

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

Bit 1 **LMF**: Limited mode flag
This bit is updated when EN bit in PKA_CR is set
- 0: All values documented in MODE bitfield can be used.
- 1: Only ECDSA verification (MODE = 0x26) is supported by the PKA.
Bit 0  **INITOK**: PKA initialization OK
- This bit is asserted when PKA initialization is complete. When RNG is not able to output proper random numbers INITOK stays at 0.
- 0: PKA is not initialized correctly. START bit cannot be set.
- 1: PKA is initialized correctly and can be used normally.

### 40.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- Bit 21  **OPERRFC**: Clear operation error flag
  - 0: No action
  - 1: Clear the OPERRF flag in PKA_SR

- Bit 20  **ADDRERRFC**: Clear address error flag
  - 0: No action
  - 1: Clear the ADDRERRF flag in PKA_SR

- Bit 19  **RAMERRFC**: Clear PKA RAM error flag
  - 0: No action
  - 1: Clear the RAMERRF flag in PKA_SR

- Bit 18 Reserved, must be kept at reset value.

- Bit 17  **PROCENDFC**: Clear PKA end of operation flag
  - 0: No action
  - 1: Clear the PROCENDF flag in PKA_SR

- Bits 16:0 Reserved, must be kept at reset value.

**Note:** Reading PKA_CLRFR returns all 0s.

### 40.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA_AHB interface.

- RAM size is 5336 bytes (max word offset: 0x14D0)

**Note:** PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 40.3.3: PKA reset and clocks.
## 40.7.5 PKA register map

### Table 401. PKA register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | PKA_CR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | PKA_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | PKA_CLRFR    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 149 for the register boundary addresses.
41 Advanced-control timers (TIM1)

In this section, “TIMx” should be understood as “TIM1” since there is only one instance of this type of timer for the products to which this reference manual applies.

41.1 TIM1 introduction

The advanced-control timer (TIM1) consists of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 41.3.30: Timer synchronization.
41.2 TIM1 main features

TIM1 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
41.3 TIM1 functional description

41.3.1 Block diagram

Figure 397. Advanced-control timer block diagram

Notes:

- **Preload registers transferred to active registers on UEV event according to control bit**
- Event
- Interrupt & DMA output

1. This feature is not available on all timers, refer to Section 41.3.2: TIM1 pins and internal signals.
2. See Figure 444: Break and Break2 circuitry overview for details.
### 41.3.2 TIM1 pins and internal signals

The tables in this section summarize the TIM inputs and outputs.

#### Table 402. TIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.</td>
</tr>
<tr>
<td>TIM_CH2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH1N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH2N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH3N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH4N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_ETR</td>
<td>Input</td>
<td>External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.</td>
</tr>
<tr>
<td>TIM_BKIN</td>
<td>Input / Output</td>
<td>Break and Break2 inputs. These inputs can also be configured in bidirectional mode.</td>
</tr>
<tr>
<td>TIM_BKIN2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table 403. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.</td>
</tr>
<tr>
<td>tim_ti2_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti3_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti4_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_etr[15:0]</td>
<td>Input</td>
<td>External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.</td>
</tr>
<tr>
<td>tim_itr[15:0]</td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_trgo/tim_trgo2</td>
<td>Output</td>
<td>Internal trigger outputs. These triggers are used by other timers and /or other peripherals.</td>
</tr>
</tbody>
</table>
Table 403. TIM internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ocref_clr[7:0]</td>
<td>Input</td>
<td>Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Input</td>
<td>Break input for internal signals</td>
</tr>
<tr>
<td>tim_brk2_cmp[8:1]</td>
<td>Input</td>
<td>Break2 input for internal signals</td>
</tr>
<tr>
<td>tim_sys_brk[n:0]</td>
<td>Input</td>
<td>System break input. This input gathers the MCU’s system level errors.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock</td>
</tr>
<tr>
<td>tim_cc_it</td>
<td>Output</td>
<td>Timer capture/compare interrupt</td>
</tr>
<tr>
<td>tim_upd_it</td>
<td>Output</td>
<td>Timer update event interrupt</td>
</tr>
<tr>
<td>tim_brk_terr_ierr_it</td>
<td>Output</td>
<td>Timer break, break2, transition error and index error interrupt</td>
</tr>
<tr>
<td>tim_trgi_com_dir_idx_it</td>
<td>Output</td>
<td>Timer trigger, commutation, direction and index interrupt</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Output</td>
<td>Timer capture / compare 1..4 dma requests</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td>Output</td>
<td>Timer trigger dma request</td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td>Output</td>
<td>Timer commutation dma request</td>
</tr>
<tr>
<td>tim_upd_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
<tr>
<td>tim_trgi_dma</td>
<td>Output</td>
<td>Timer trigger dma request</td>
</tr>
<tr>
<td>tim_com_dma</td>
<td>Output</td>
<td>Timer commutation dma request</td>
</tr>
</tbody>
</table>

Table 404, Table 405, Table 406 and Table 407 list the sources connected to the tim_ti[4:1] input multiplexers.

Table 404. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in0</td>
<td>TIM1</td>
</tr>
<tr>
<td>tim_ti1_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 405. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti2 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti2_in0</td>
<td>TIM1</td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 406. Interconnect to the tim_ti3 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti3 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti3_in0</td>
<td>TIM1</td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td>TIM1_CH3</td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 407. Interconnect to the tim_ti4 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti4 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti4_in0</td>
<td>TIM1</td>
</tr>
<tr>
<td>tim_ti4_in[15:1]</td>
<td>TIM1_CH4</td>
</tr>
<tr>
<td>tim_ti4_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 408 lists the internal sources connected to the tim_itr input multiplexer.

Table 408. Internal trigger connection

<table>
<thead>
<tr>
<th>Timer internal trigger input signal</th>
<th>TIM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>tim_itr4</td>
<td>tim5_trgo</td>
</tr>
<tr>
<td>tim_itr5</td>
<td>tim9_trgo</td>
</tr>
<tr>
<td>tim_itr6</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>tim_itr7</td>
<td>tim13_oc1</td>
</tr>
<tr>
<td>tim_itr8</td>
<td>tim14_oc1</td>
</tr>
<tr>
<td>tim_itr9</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>tim_itr10</td>
<td>tim16_oc1</td>
</tr>
<tr>
<td>tim_itr11</td>
<td>tim17_oc1</td>
</tr>
<tr>
<td>tim_itr[15:12]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 409 lists the internal sources connected to the tim_etr input multiplexer.

Table 409. Interconnect to the tim_etr input multiplexer

<table>
<thead>
<tr>
<th>Timer external trigger input signal</th>
<th>Timer external trigger signals assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_etr0</td>
<td>TIM1_ETR</td>
</tr>
<tr>
<td>tim_etr[2:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 409. Interconnect to the tim_etr input multiplexer (continued)

<table>
<thead>
<tr>
<th>Timer external trigger input signal</th>
<th>Timer external trigger signals assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_etr3</td>
<td>adc1_awd1</td>
</tr>
<tr>
<td>tim_etr4</td>
<td>adc1_awd2</td>
</tr>
<tr>
<td>tim_etr5</td>
<td>adc1_awd3</td>
</tr>
<tr>
<td>tim_etr6</td>
<td>adc2_awd1</td>
</tr>
<tr>
<td>tim_etr7</td>
<td>adc2_awd2</td>
</tr>
<tr>
<td>tim_etr8</td>
<td>adc2_awd3</td>
</tr>
<tr>
<td>tim_etr[15:9]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 410, Table 411 and Table 412 list the sources connected to the tim_brk and tim_brk2 inputs.

Table 410. Timer break interconnect

<table>
<thead>
<tr>
<th>TIM_Bkin inputs</th>
<th>TIM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_Bkin</td>
<td>TIM1_Bkin pin</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 411. Timer break2 interconnect

<table>
<thead>
<tr>
<th>TIM_Bkin2 inputs</th>
<th>TIM1</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_Bkin2</td>
<td>TIM1_Bkin2 pin</td>
</tr>
<tr>
<td>tim_brk2_cmp[8:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 412. System break interconnect

<table>
<thead>
<tr>
<th>TIM1/TIM8</th>
<th>Enable bit in SBS_BRK_LOCKUP register</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_sys_brk0</td>
<td>AXISRAM1 double ECC error</td>
</tr>
<tr>
<td>tim_sys_brk1</td>
<td>AXISRAM3 double ECC error</td>
</tr>
<tr>
<td>tim_sys_brk2</td>
<td>ITCM double ECC error</td>
</tr>
<tr>
<td>tim_sys_brk3</td>
<td>DTCM double ECC error</td>
</tr>
<tr>
<td>tim_sys_brk4</td>
<td>Backup RAM double ECC error</td>
</tr>
<tr>
<td>tim_sys_brk5</td>
<td>Cortex-M7 LOCKUP</td>
</tr>
<tr>
<td>tim_sys_brk6</td>
<td>FLASH double ECC error</td>
</tr>
<tr>
<td>tim_sys_brk7</td>
<td>Programmable Voltage Detector (PVD)</td>
</tr>
<tr>
<td>CSS</td>
<td>Clock Security System</td>
</tr>
</tbody>
</table>
41.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 398* and *Figure 399* give some examples of the counter behavior when the prescaler ratio is changed on the fly.
Figure 398. Counter timing diagram with prescaler division change from 1 to 2

```
<table>
<thead>
<tr>
<th>tim_psc_ck</th>
<th>CEN</th>
<th>tim_cnt_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Counter register: F7 F8 F9 FA FB FC 00 01 02 03

Update event (UEV)

Prescaler control register: 0 1

Write a new value in TIMx_PSC

Prescaler buffer: 0 1 0 1 0 1 0 1

Prescaler counter: 0 0 1 0 1 0 1 0 1

Figure 399. Counter timing diagram with prescaler division change from 1 to 4

```
<table>
<thead>
<tr>
<th>tim_psc_ck</th>
<th>CEN</th>
<th>tim_cnt_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Counter register: F7 F8 F9 FA FB FC 00 01

Update event (UEV)

Prescaler control register: 0 1 3

Write a new value in TIMx_PSC

Prescaler buffer: 0 3

Prescaler counter: 0 0 1 2 3 0 1 2 3
41.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.
**Figure 400. Counter timing diagram, internal clock divided by 1**

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register:**
  - 31 32 33 34 35 36 00 01 02 03 04 05 06 07
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

**Figure 401. Counter timing diagram, internal clock divided by 2**

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register:**
  - 0034 0035 0036 0000 0001 0002 0003
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
Figure 402. Counter timing diagram, internal clock divided by 4

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register**: 0035, 0036, 0000, 0001
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

Figure 403. Counter timing diagram, internal clock divided by N

- **tim_psc_ck**
- **tim_cnt_ck**
- **Counter register**: 1F, 20, 00
- **Counter overflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
Figure 404. Counter timing diagram, update event when ARPE = 0
(TIMx_ARR not preloaded)

<table>
<thead>
<tr>
<th>Counter register</th>
<th>31</th>
<th>32</th>
<th>33</th>
<th>34</th>
<th>35</th>
<th>36</th>
<th>00</th>
<th>01</th>
<th>02</th>
<th>03</th>
<th>04</th>
<th>05</th>
<th>06</th>
<th>07</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counter overflow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update event (UEV)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update interrupt flag (UIF)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auto-reload preload register</td>
<td>FF</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Write a new value in TIMx_ARR
**Downcounting mode**

In downcounting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn’t change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

**Figure 406. Counter timing diagram, internal clock divided by 1**
Figure 407. Counter timing diagram, internal clock divided by 2

Figure 408. Counter timing diagram, internal clock divided by 4
Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the...
autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.
Figure 411. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6

1. Here, center-aligned mode 1 is used (for more details refer to Section 41.6: TIM1 registers).

Figure 412. Counter timing diagram, internal clock divided by 2
**Figure 413. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36**

- `tim_psc_ck`
- `tim_cnt_ck`
- CEN
- Counter register: 0034 0035 0036 0035
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

Note: Here, center aligned mode 2 or 3 is updated with an UIF on overflow

**Figure 414. Counter timing diagram, internal clock divided by N**

- `tim_psc_ck`
- `tim_cnt_ck`
- Counter register: 20 1F 01 00
- Counter underflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 415. Counter timing diagram, update event with ARPE = 1 (counter underflow)
41.3.5 Repetition counter

Section 41.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode.

Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is 2xTck due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 417). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

**Figure 417. Update rate examples depending on mode and TIMx_RCR register settings**

<table>
<thead>
<tr>
<th>RCR Value</th>
<th>Counter-aligned mode</th>
<th>Edge-aligned mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><img src="image" alt="Counter-aligned mode" /></td>
<td><img src="image" alt="Upcounting" /></td>
</tr>
<tr>
<td>1</td>
<td><img src="image" alt="Counter-aligned mode" /></td>
<td><img src="image" alt="Upcounting" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Counter-aligned mode" /></td>
<td><img src="image" alt="Upcounting" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image" alt="Counter-aligned mode" /></td>
<td><img src="image" alt="Upcounting" /></td>
</tr>
<tr>
<td>3 and re-synchronization</td>
<td><img src="image" alt="Counter-aligned mode" />(by SW)</td>
<td><img src="image" alt="Upcounting" />(by SW)</td>
</tr>
</tbody>
</table>

**Legend:**
- UEV: Update event
- (by SW): Preload registers transferred to active registers and update interrupt generated
- Update Event: if the repetition counter underflow occurs when the counter is equal to the auto-reload value

### 41.3.6 External trigger input

The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see Section 41.3.7)
- trigger for the slave mode (see Section 41.3.30)
- PWM reset input for cycle-by-cycle current regulation (see Section 41.3.9)

**Figure 418** below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition...
the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

**Figure 418. External trigger input block**

The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 41.3.2: TIM1 pins and internal signals* for the list of sources connected to the etr_in input in the product.

### 41.3.7 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

**Internal clock source (tim_ker_ck)**

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

*Figure 419* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
**Figure 419. Control circuit in normal mode, internal clock divided by 1**

- `tim_ker_ck`
- `CEN`
- `UG`
- `counter initialization (internal)`
- `tim_cnt_ck, tim_psc_ck`
- `Counter register`

**External clock source mode 1**

This mode is selected when `SMS = 111` in the `TIMx_SMCR` register. The counter can count at each rising or falling edge on a selected input.

**Figure 420. tim_ti2 external clock connection example**

1. Codes ranging from `01000` to `11111` are reserved.
For example, to configure the upcounter to count in response to a rising edge on the `tim_ti2` input, use the following procedure:

1. Configure channel 2 to detect rising edges on the `tim_ti2` input by writing `CC2S = 01` in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the `IC2F[3:0]` bits in the TIMx_CCMR1 register (if no filter is needed, keep `IC2F = 0000`).
3. Select rising edge polarity by writing `CC2P = 0` and `CC2NP = 0` in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing `SMS = 111` in the TIMx_SMCR register.
5. Select `tim_ti2` as the trigger input source by writing `TS = 00110` in the TIMx_SMCR register.
6. Enable the counter by writing `CEN = 1` in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on `tim_ti2`, the counter counts once and the TIF flag is set. The delay between the rising edge on `tim_ti2` and the actual clock of the counter is due to the resynchronization circuit on `tim_ti2` input.

**Figure 421. Control circuit in external clock mode 1**

**External clock source mode 2**

This mode is selected by writing `ECE = 1` in the TIMx_SMCR register. The counter counts at each rising or falling edge on the external trigger input `tim_etr_in`. The **Figure 422** gives an overview of the external trigger input block.
For example, to configure the upcounter to count each 2 rising edges on \texttt{tim\_etr\_in}, use the following procedure:

1. As no filter is needed in this example, write \texttt{ETF[3:0] = 0000} in the \texttt{TIMx\_SMCR} register.
2. Set the prescaler by writing \texttt{ETPS[1:0] = 01} in the \texttt{TIMx\_SMCR} register.
3. Select rising edge detection on the \texttt{tim\_etr\_in} input by writing \texttt{ETP = 0} in the \texttt{TIMx\_SMCR} register.
4. Enable external clock mode 2 by writing \texttt{ECE = 1} in the \texttt{TIMx\_SMCR} register.
5. Enable the counter by writing \texttt{CEN = 1} in the \texttt{TIMx\_CR1} register.

The counter counts once each 2 \texttt{tim\_etr\_in} rising edges.

The delay between the rising edge on \texttt{tim\_etr\_in} and the actual clock of the counter is due to the resynchronization circuit on the \texttt{tim\_etr} signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most \(1/4\) of \texttt{tim\_ker\_ck} frequency. When the \texttt{ETRP} signal is faster, the user must apply a division of the external signal by a proper \texttt{ETPS} prescaler setting.

---

1. Refer to Section 41.3.2: TIM1 pins and internal signals.
41.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

*Figure 424* to *Figure 427* give an overview of one capture/compare channel.

The input stage samples the corresponding `tim_tix` input to generate a filtered signal `tim_tixf`. Then, an edge detector with polarity selection generates a signal (`tim_tixfpy`) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

*Figure 424. Capture/compare channel (example: channel 1 input stage)*
The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

**Figure 425. Capture/compare channel 1 main circuit**

**Figure 426. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)**

1. tim_ocxref, where x is the rank of the complementary channel
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 41.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.

- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level
have been detected (sampled at $f_{DTS}$ frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the tim_t11 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

**Note:** IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

### 41.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges).

This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.
The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

![Figure 428. PWM input mode timing](image)

**41.3.11 Forced output mode**

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.
Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

### 41.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCMRx register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

#### Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 429.
41.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per TIMx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSl, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.
PWM edge-aligned mode

- Upcounting configuration
  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to Upcounting mode.
  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. Figure 430 shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

  Figure 430. Edge-aligned PWM waveforms (ARR = 8)

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCRx=4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx&gt;8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>‘1’</td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCRx=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>‘0’</td>
<td></td>
</tr>
<tr>
<td>tim_ocxref</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCxIF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Downcounting configuration
  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the Downcounting mode.
  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit
(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to Center-aligned mode (up/down counting).

Figure 431 shows some center-aligned PWM waveforms in an example where:
- TIMx_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 431. Center-aligned PWM waveforms (ARR = 8)

Hints on using center-aligned mode:
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit.
in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  -- The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  -- The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. **Figure 432** presents the dithering principle applied to four consecutive PWM cycles.

**Figure 432. Dithering principle**

When the dithering mode is enabled, the register coding is changed as follows (see **Figure 433** for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

**Note:** The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).
The minimum frequency is given by the following formula:

\[ \text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwm Min}} = \frac{F_{\text{Tim Max}}}{\text{Resolution}} \]

Dithering mode disabled: \( F_{\text{pwm Min}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode enabled: \( F_{\text{pwm Min}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \)

**Note:** The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on Figure 434, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.
The duty cycle and/or period changes are spread over 16 consecutive periods, as described in Figure 435.
The autoreload and compare values increments are spread following specific patterns described in Table 413. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 413. CCR and ARR register change dithering pattern

<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0000</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
</tr>
</tbody>
</table>
The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in Figure 436.

Table 414 shows how the dithering pattern is added in center-aligned PWM mode.

```
<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td>
</tr>
<tr>
<td>0000</td>
<td>- - - - - - - - - - - - - - - - -</td>
</tr>
<tr>
<td>0001</td>
<td>- - - - - - - - - - - - - - - - -</td>
</tr>
<tr>
<td>0010</td>
<td>+1 - - - - - - - - - - - - - - - -</td>
</tr>
<tr>
<td>0011</td>
<td>+1 - - - - +1 - - - - - - - - - -</td>
</tr>
<tr>
<td>0100</td>
<td>+1 - - - +1 - - - - - - - - +1 - -</td>
</tr>
<tr>
<td>0101</td>
<td>+1 - +1 - - - - - - - - - - - - -</td>
</tr>
<tr>
<td>0110</td>
<td>+1 - +1 - +1 - - - - - - - - - -</td>
</tr>
<tr>
<td></td>
<td>Up  Dn  Up  Dn  Up  Dn  Up  Dn  Up  Dn  Up  Dn  Up  Dn  Up  Dn</td>
</tr>
</tbody>
</table>
```

Table 413. CCR and ARR register change dithering pattern (continued)

```
<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>1011</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
<tr>
<td>1010</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
<tr>
<td>1100</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
<tr>
<td>1101</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
<tr>
<td>1110</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
<tr>
<td>1111</td>
<td>+1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1 +1</td>
</tr>
</tbody>
</table>
```

Table 414. CCR register change dithering pattern in center-aligned PWM mode
41.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2refc signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 437 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.
41.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, \( \text{tim\_ocxrefc} \), are made of an OR or AND logical combination of two reference PWMs:

- \( \text{tim\_oc1refc} \) (or \( \text{tim\_oc2refc} \)) is controlled by TIMx_CCR1 and TIMx_CCR2
- \( \text{tim\_oc3refc} \) (or \( \text{tim\_oc4refc} \)) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one \( \text{tim\_ocx} \) output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

*Note:* The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

*Figure 438* represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.
41.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5.
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5.
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5.

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].
The *tim_trgo2* waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to Section 41.3.31: ADC triggers for more details.

### 41.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output *tim_ocx* or complementary *tim_ocxn*) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals *tim_ocx* and *tim_ocxn* are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 422: Output control bits for complementary *tim_ocx* and *tim_ocxn* channels with break feature for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

**Figure 440. Complementary output with symmetrical dead-time insertion**

The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on Figure 441.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

**Note:** If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.
The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 41.6.20: TIM1 break and dead-time register (TIM1_BDTR) for delay calculation.
Redirecting `tim_ocxref` to `tim_ocx` or `tim_ocxn`

In output mode (forced, output compare or PWM), `tim_ocxref` can be redirected to the `tim_ocx` output or to `tim_ocxn` output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

**Note:** When only `tim_ocxn` is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as `tim_ocxref` is high. For example, if CCxNP = 0 then `tim_ocxn = tim_ocxref`. On the other hand, when both `tim_ocx` and `tim_ocxn` are enabled (CCxE = CCxNE = 1) `tim_ocx` becomes active when `tim_ocxref` is high whereas `tim_ocxn` is complemented and becomes active when `tim_ocxref` is low.

### 41.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The `tim_ocx` and `tim_ocxn` outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 422: Output control bits for complementary `tim_ocx` and `tim_ocxn` channels with break feature for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous
and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to Section 41.3.2: TIM1 pins and internal signals for product specific implementation)
  - coming from a system break request (refer to Section 41.3.2: TIM1 pins and internal signals for product specific implementation)

The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to Section 41.3.2: TIM1 pins and internal signals for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per Figure 444 below.
Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.
their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

- If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF, and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.

- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to Section 41.6.20: TIM1 break and dead-time register (TIM1_BDTR). The LOCK bits can be written only once after an MCU reset.

Figure 445 shows an example of behavior of the outputs in response to a break.
The two break inputs have different behaviors on timer outputs:

- The `tim_brk` input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- `tim_brk2` can only disable (inactive state) the PWM outputs.
The `tim_brk` has a higher priority than `tim_brk2` input, as described in Table 415.

**Note:** `tim_brk2` must only be used with OSSR = OSSI = 1.

<table>
<thead>
<tr>
<th><code>tim_brk</code></th>
<th><code>tim_brk2</code></th>
<th>Timer outputs state</th>
<th>Typical use case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>X</td>
<td>– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)</td>
<td>ON after deadtime insertion</td>
</tr>
<tr>
<td>Inactive</td>
<td>Active</td>
<td>Inactive</td>
<td>OFF</td>
</tr>
</tbody>
</table>

**Table 415. Behavior of timer outputs versus `tim_brk`/`tim_brk2` inputs**

**Figure 446** gives an example of `tim_ocx` and `tim_ocxn` output behavior in case of active signals on `tim_brk` and `tim_brk2` inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

**Figure 446. PWM output state following `tim_brk` and `tim_brk2` assertion (OSSI = 1)**
41.3.19 Bidirectional break inputs

The TIM1 features bidirectional break I/Os, as represented on Figure 448.

This provides support for:
- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:
- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 416).
Arming and rearming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKBID (BK2BID)</th>
<th>BKDSRM (BK2DSRM)</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

### Table 416. Break protection disarming conditions

**Figure 448. Output redirection (tim_brk2 request not represented)**

---

**41.3.20 Clearing the tim_ocxref signal on an external event**

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocxref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM
cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

tim_etr_in must be configured as follows:
1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

*Figure 449* shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

**Figure 449. Clearing TIMx tim_ocxref**

![Figure 449](MSv62342V1)

**Note:** In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

### 41.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).
A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 450 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

**41.3.22 One-pulse mode**

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.
A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

**Figure 451. Example of one pulse mode.**

In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The tDELAY is defined by the value written in the TIMx_CCR1 register.
- The tPULSE is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the
TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low. Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay \( t_{\text{DELAY min}} \) that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 41.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in Section 41.3.22:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones. This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.
41.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on Figure 453.

**Figure 453. Pulse generator circuitry**

*Figure 454* shows how the pulse is generated for edge-aligned and encoder operating modes.
This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

\[ t_{PW} = PW[7:0] \times t_{PWG} \]

where \( t_{PWG} = \left(2^{(PWPRSC[2:0])}\right) \times t_{tim\_ker\_ck} \)

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 455).
41.3.25 Encoder interface mode

**Quadrature encoder**

To select Encoder Interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to a quadrature encoder. Refer to Table 417. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.
A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

*Figure 456* gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

### Table 417. Counting direction versus encoder signals (CC1P = CC2P = 0)

<table>
<thead>
<tr>
<th>Active edge</th>
<th>SMS[3:0]</th>
<th>Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)</th>
<th>tim_ti1fp1 signal</th>
<th>tim_ti2fp2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting on tim_ti1 only x1 mode</td>
<td>1110</td>
<td>High, Down, Up, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, No count, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>Counting on tim_ti2 only x1 mode</td>
<td>1111</td>
<td>High, No count, No count, Up, Down</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, No count, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>Counting on tim_ti1 only x2 mode</td>
<td>0001</td>
<td>High, Down, Up, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, Up, Down, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>Counting on tim_ti2 only x2 mode</td>
<td>0010</td>
<td>High, No count, No count, Up, Down</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, No count, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>Counting on tim_ti1 and tim_ti2</td>
<td>0011</td>
<td>High, Down, Up, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
<tr>
<td>x4 mode</td>
<td></td>
<td>Low, Up, Down, No count, No count</td>
<td>Rising</td>
<td>Falling</td>
</tr>
</tbody>
</table>

In this table, each row represents a different configuration of the counter. The columns represent the level on the opposite signal and the signal on tim_ti1fp1 and tim_ti2fp2. The final columns indicate the counting direction for each edge.
Figure 456. Example of counter operation in encoder interface mode.

Figure 457 gives an example of counter behavior when tim_token1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 458 shows the timer counter value during a speed reversal, for various counting modes.
The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register’s bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

### Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

In the clock plus direction mode shown on Figure 459, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- **1010**: x2 mode, the counter is updated on both rising and falling edges of the clock
- **1011**: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

**Figure 458. Quadrature encoder counting modes**

![Diagram of quadrature encoder counting modes](image-url)
The polarity of the direction signal on tim_t1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_t1 is high and down-counting when tim_t1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_t1 is low).

**Figure 459. Direction plus clock encoder mode**

![Directional clock encoder mode diagram](MSv62352V1)

**Directional clock encoder mode**

In the directional clock mode on Figure 460, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to Figure 460) and CCxP = 1 corresponds to low-level idle state (refer to Figure 461).

- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to Figure 460), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to Figure 461).

**Figure 460. Directional clock encoder mode (CC1P = CC2P = 0)**

![Directional clock encoder mode diagram](MSv62353V1)
Figure 461. Directional clock encoder mode (CC1P = CC2P = 1)

Table 418 here-below details how the directional clock mode operates, for any input transition.

Table 418. Counting direction versus encoder signals and polarity settings

<table>
<thead>
<tr>
<th>Directional clock mode</th>
<th>SMS[3:0]</th>
<th>Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)</th>
<th>tim_ti1fp1 signal</th>
<th>tim_ti2fp2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>tim_ti1fp1 signal</td>
<td>tim_ti2fp2 signal</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>x2 mode CCxP = 0</td>
<td>1100</td>
<td>High</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x2 mode CCxP = 1</td>
<td>1100</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>x1 mode CCxP = 0</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x1 mode CCxP = 1</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>No count</td>
</tr>
</tbody>
</table>

Index input

The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.
Available encoders are proposed with several options for index pulse conditioning, as per Figure 462:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

**Figure 462. Index gating options**

The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on Figure 463.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

**Figure 463. Jittered Index signals**

The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and
channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. Figure 464 shows at which position is the index generated, for a simplistic example (an encoder providing four edges per mechanical rotation).

**Figure 464. Index generation for IPOS[1:0] = 11**

---

**Figure 465** presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

**Figure 465. Counter reading with index gated on channel A (IPOS[1:0] = 11)**
Figure 466. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

**Figure 466. Counter reading with index ungated (IPOS[1:0] = 00)**

Figure 467. shows how the ‘gated on A & B’ mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

**Figure 467. Counter reading with index gated on channel A and B**

Figure 468 and Figure 469 detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.
Figure 468. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

Channel A

Channel B

Index

DIR bit

Counter

Index leading state transition

Index delayed versus state transition
Figure 469. Counter reset Narrow index pulse (closer view, ARR = 0x07)
Figure 470 shows how the index is managed in x1 and x2 modes.

**Figure 470. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)**

![Diagram showing index behavior in x1 and x2 mode](MSv45773V1)

**Directional index sensitivity**

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 471 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

**Figure 471. Directional index sensitivity**

![Diagram showing directional index sensitivity](MSv45774V1)
Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on Figure 472. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

**Figure 472. Counter reset as function of FIDX bit setting**

Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the Figure 473 below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

**Figure 473. Index blanking**
Index management in nonquadrature mode

*Figure 474* and *Figure 475* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

### Figure 474. Index behavior in clock + direction mode, IPOS[0] = 1

![Figure 474](image1)

### Figure 475. Index behavior in directional clock mode, IPOS[0] = 1

![Figure 475](image2)

Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 476*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR...
status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 476. State diagram for quadrature encoded signals

For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on Figure 477.

**Figure 477. Up-counting encoder error detection**
In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on Figure 478, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

**Figure 478. Down-counting encode error detection**

An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

**Functional encoder interrupts**

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.

- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on Figure 479.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 479. Encoder mode change with preload transferred on update (SMSPS = 0)

Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

41.3.26 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.
This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

### 41.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register’s bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flags assertion.

### 41.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins TIMt1, TIMt2 and TIMt3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per Figure 480.

*Figure 480. Measuring time interval between edges on three signals*

### 41.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as “interfacing timer” in Figure 481. The “interfacing timer” captures the three timer input pins (TIMt1, TIMt2 and TIMt3) connected through a XOR to the TIMt1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is TIMt1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the “interfacing timer”, capture/compare channel 1 is configured in capture mode, capture signal is TIM_trc (See Figure 424). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.
The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to 1.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIMx_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to 101.

In the advanced-control timer, the right tim_itr0 input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 481 describes this example.
### 41.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 42.4.23: Timer synchronization for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 0010 in TIMx_SMCR register.
- Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 0010 in TIMx_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).
The counter starts counting on the internal clock as long as tim_t1 is low and stops as soon as tim_t1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_t1 and the actual stop of the counter is due to the resynchronization circuit on tim_t1 input.

**Slave mode: Trigger mode**

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on tim_t2 input:

- Configure the channel 2 to detect rising edges on tim_t2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_t2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_t2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_t2 and the actual start of the counter is due to the resynchronization circuit on tim_t2 input.
Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter. This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled. This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01 in TIMx_CCMR1 register to select only the input capture source
   - CC1NP = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register.
   Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

   A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

**Figure 485. Control circuit in external clock mode 2 + trigger mode**

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
41.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in Figure 439.

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

41.3.32 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see Section 41.6.29: TIM1 DMA control register (TIM1_DCR) for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.
This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (see note below).
   - Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bitfields as follows:
   - DBL = 3 transfers, DBA = 0xE and DBSS = 1.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx.

5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let’s take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

**Note:** A null value can be written to the reserved registers.

### 41.3.33 TIM1 DMA requests

The TIM1 can generate a DMA request, as shown in the table below.

<table>
<thead>
<tr>
<th>DMA request signal</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Capture/compare 1</td>
<td>CC1DE</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Capture/compare 2</td>
<td>CC2DE</td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td>Capture/compare 3</td>
<td>CC3DE</td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td>Capture/compare 4</td>
<td>CC4DE</td>
</tr>
<tr>
<td>tim_com_dma</td>
<td>Commutation (COM)</td>
<td>COMDE</td>
</tr>
<tr>
<td>tim_trgi_dma</td>
<td>Trigger</td>
<td>TDE</td>
</tr>
</tbody>
</table>

### 41.3.34 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.
For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.
For more details, refer to section Debug support (DBG).

41.4 TIM1 low-power modes

Table 420. Effect of low-power modes on TIM1

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

41.5 TIM1 interrupts

The TIM1 can generate multiple interrupts, as shown in Table 421.

Table 421. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_UP</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_CC</td>
<td>Capture/compare 1</td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 2</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 3</td>
<td>CC3IF</td>
<td>CC3IE</td>
<td>write 0 in CC3IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 4</td>
<td>CC4IF</td>
<td>CC4IE</td>
<td>write 0 in CC4IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TRG_COM</td>
<td>Commutation (COM)</td>
<td>COMIF</td>
<td>COMIE</td>
<td>write 0 in COMIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Trigger</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_DIR_IDX</td>
<td>Index</td>
<td>IDXF</td>
<td>IDXIE</td>
<td>write 0 in IDXF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Direction</td>
<td>DIRF</td>
<td>DIRIE</td>
<td>write 0 in DIRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_BRK</td>
<td>Break</td>
<td>BIF</td>
<td>BIE</td>
<td>write 0 in BIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Break2</td>
<td>B2IF</td>
<td>BIE</td>
<td>write 0 in B2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>System Break</td>
<td>SBIF</td>
<td></td>
<td>write 0 in SBIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_IERR</td>
<td>Index Error</td>
<td>IERRF</td>
<td>IERRIE</td>
<td>write 0 in IERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TER</td>
<td>Transition Error</td>
<td>TERRF</td>
<td>TERRIE</td>
<td>write 0 in TERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
41.6 TIM1 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

41.6.1 TIM1 control register 1 (TIM1_CR1)

Address offset: 0x000

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
- 0: Dithering disabled
- 1: Dithering enabled

*Note: The DITHEN bit can only be modified when CEN bit is reset.*

Bit 11 **UIFREMAP**: UIF status bit remapping
- 0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
- 1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_etru_in, tim_fix),
- 00: tDTS = ttim_ker_ck
- 01: tDTS = 2*ttim_ker_ck
- 10: tDTS = 4*ttim_ker_ck
- 11: Reserved, do not program this value

Bit 7 **ARPE**: Autoreload preload enable
- 0: TIMx_ARR register is not buffered
- 1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection
- 00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
- 01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.
- 10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.
- 11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

*Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)*
Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter

*Note:* This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value
(ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is
set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

*Note:* External clock, gated mode and encoder mode can work only if the CEN bit has been
previously set by software. However trigger mode can set the CEN bit automatically by hardware.

### 41.6.2 TIM1 control register 2 (TIM1_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OIS4N</td>
<td>OIS4</td>
<td>OIS3N</td>
<td>OIS3</td>
<td>OIS2N</td>
<td>OIS2</td>
<td>OIS1N</td>
<td>OIS1</td>
<td>TI1S</td>
<td>MMS[2:0]</td>
<td>CCDS</td>
<td>CCUS</td>
<td>CCPC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

- **0000**: *Reset* - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.
- **0001**: *Enable* - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).
- **0010**: *Update* - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.
- **0011**: *Compare pulse* - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).
- **0100**: *Compare* - tim_oc1refc signal is used as trigger output (tim_trgo2)
- **0101**: *Compare* - tim_oc2refc signal is used as trigger output (tim_trgo2)
- **0110**: *Compare* - tim_oc3refc signal is used as trigger output (tim_trgo2)
- **0111**: *Compare* - tim_oc4refc signal is used as trigger output (tim_trgo2)
- **1000**: *Compare* - tim_oc5refc signal is used as trigger output (tim_trgo2)
- **1001**: *Compare* - tim_oc6refc signal is used as trigger output (tim_trgo2)
- **1010**: *Compare Pulse* - tim_oc4refc rising or falling edges generate pulses on tim_trgo2
- **1011**: *Compare pulse* - tim_oc6refc rising or falling edges generate pulses on tim_trgo2
- **1100**: *Compare pulse* - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2
- **1101**: *Compare pulse* - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2
- **1110**: *Compare pulse* - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2
- **1111**: *Compare pulse* - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

**Note**: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)
Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit
Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)  
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)  
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)  
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)  
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)  
0: tim_oc1n = 0 after a dead-time when MOE = 0  
1: tim_oc1n = 1 after a dead-time when MOE = 0  
**Note**: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)  
0: tim_oc1 = 0 (after a dead-time) when MOE = 0  
1: tim_oc1 = 1 (after a dead-time) when MOE = 0  
**Note**: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection  
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input  
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input
Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

- **0000**: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

- **0001**: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

- **0010**: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

- **0011**: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

- **0100**: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

- **0101**: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

- **0110**: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

- **0111**: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

- **1000**: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

*Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.*

Bit 3 **CCDS**: Capture/compare DMA selection

- 0: CCx DMA request sent when CCx event occurs
- 1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

- 0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only
- 1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

- 0: CCxE, CCxNE and OCxM bits are not preloaded
- 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*
41.6.3 TIM1 slave mode control register (TIM1_SMCR)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 25 **SMSPS**: SMS preload source
This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active
0: The transfer is triggered by the Timer’s Update event
1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable
This bit selects whether the SMS[3:0] bitfield is preloaded
0: SMS[3:0] bitfield is not preloaded
1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3
Refer to TS[2:0] description - bits 6:4

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity
This bit selects whether tim_etr_in or tim_etr_in is used for trigger operations
0: tim_etr_in is non-inverted, active at high level or rising edge.
1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable
This bit enables External clock mode 2.
0: External clock mode 2 disabled
1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

**Note**: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.
Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at fDTS
0001: fSAMPLING = ftim_ker_ck, N = 2
0010: fSAMPLING = ftim_ker_ck, N = 4
0011: fSAMPLING = ftim_ker_ck, N = 8
0100: fSAMPLING = fDTS/2, N = 6
0101: fSAMPLING = fDTS/2, N = 8
0110: fSAMPLING = fDTS/4, N = 6
0111: fSAMPLING = fDTS/4, N = 8
1000: fSAMPLING = fDTS/8, N = 6
1001: fSAMPLING = fDTS/8, N = 8
1010: fSAMPLING = fDTS/16, N = 5
1011: fSAMPLING = fDTS/16, N = 6
1100: fSAMPLING = fDTS/16, N = 8
1101: fSAMPLING = fDTS/32, N = 5
1110: fSAMPLING = fDTS/32, N = 6
1111: fSAMPLING = fDTS/32, N = 8

Bit 7 **MSM**: Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.
Bits 6:4  **TS[2:0]: Trigger selection**

This bitfield is combined with TS[4:3] bits. This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See Table 408: Internal trigger connection for more details on tim_itrx meaning for each Timer.

**Note:** These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3  Reserved, must be kept at reset value.
Bits 16, 2:0 **SMS[3:0]**: Slave mode selection

When external signals are selected, the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

**Note:** The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on Ti1F, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
## TIM1 DMA/interrupt enable register (TIM1_DIER)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 23    | **TERRIE**: Transition error interrupt enable  
            | 0: Transition error interrupt disabled  
            | 1: Transition error interrupt enabled |
| Bit 22    | **IERRIE**: Index error interrupt enable  
            | 0: Index error interrupt disabled  
            | 1: Index error interrupt enabled |
| Bit 21    | **DIRIE**: Direction change interrupt enable  
            | 0: Direction Change interrupt disabled  
            | 1: Direction Change interrupt enabled |
| Bit 20    | **IDXIE**: Index interrupt enable  
            | 0: Index interrupt disabled  
            | 1: Index Change interrupt enabled |
| Bit 19:15 | Reserved, must be kept at reset value. |
| Bit 14    | **TDE**: Trigger DMA request enable  
            | 0: Trigger DMA request disabled  
            | 1: Trigger DMA request enabled |
| Bit 13    | **COMDE**: COM DMA request enable  
            | 0: COM DMA request disabled  
            | 1: COM DMA request enabled |
| Bit 12    | **CC4DE**: Capture/compare 4 DMA request enable  
            | 0: CC4 DMA request disabled  
            | 1: CC4 DMA request enabled |
| Bit 11    | **CC3DE**: Capture/compare 3 DMA request enable  
            | 0: CC3 DMA request disabled  
            | 1: CC3 DMA request enabled |
| Bit 10    | **CC2DE**: Capture/compare 2 DMA request enable  
            | 0: CC2 DMA request disabled  
            | 1: CC2 DMA request enabled |
| Bit 9     | **CC1DE**: Capture/compare 1 DMA request enable  
            | 0: CC1 DMA request disabled  
            | 1: CC1 DMA request enabled |
Bit 8 **UDE**: Update DMA request enable
- 0: Update DMA request disabled
- 1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
- 0: Break interrupt disabled
- 1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
- 0: Trigger interrupt disabled
- 1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
- 0: COM interrupt disabled
- 1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
- 0: CC4 interrupt disabled
- 1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
- 0: CC3 interrupt disabled
- 1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
- 0: CC2 interrupt disabled
- 1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
- 0: Update interrupt disabled
- 1: Update interrupt enabled

### 41.6.5 TIM1 status register (TIM1_SR)

**Address offset**: 0x010

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Bit 2</th>
<th>Bit 3</th>
<th>Bit 4</th>
<th>Bit 5</th>
<th>Bit 6</th>
<th>Bit 7</th>
<th>Bit 8</th>
<th>Bit 9</th>
<th>Bit 10</th>
<th>Bit 11</th>
<th>Bit 12</th>
<th>Bit 13</th>
<th>Bit 14</th>
<th>Bit 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIF</td>
<td>COMIF</td>
<td>CC1IF</td>
<td>CC2IF</td>
<td>CC3IF</td>
<td>CC4IF</td>
<td>IERRF</td>
<td>DIF</td>
<td>IERF</td>
<td>TF</td>
<td>COMIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>B2IF</td>
</tr>
<tr>
<td>UIF</td>
<td>COMIF</td>
<td>CC1IF</td>
<td>CC2IF</td>
<td>CC3IF</td>
<td>CC4IF</td>
<td>IERRF</td>
<td>DIF</td>
<td>IERF</td>
<td>TF</td>
<td>COMIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>B2IF</td>
</tr>
<tr>
<td>UIF</td>
<td>COMIF</td>
<td>CC1IF</td>
<td>CC2IF</td>
<td>CC3IF</td>
<td>CC4IF</td>
<td>IERRF</td>
<td>DIF</td>
<td>IERF</td>
<td>TF</td>
<td>COMIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>B2IF</td>
</tr>
<tr>
<td>UIF</td>
<td>COMIF</td>
<td>CC1IF</td>
<td>CC2IF</td>
<td>CC3IF</td>
<td>CC4IF</td>
<td>IERRF</td>
<td>DIF</td>
<td>IERF</td>
<td>TF</td>
<td>COMIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>B2IF</td>
</tr>
<tr>
<td>UIF</td>
<td>COMIF</td>
<td>CC1IF</td>
<td>CC2IF</td>
<td>CC3IF</td>
<td>CC4IF</td>
<td>IERRF</td>
<td>DIF</td>
<td>IERF</td>
<td>TF</td>
<td>COMIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>B2IF</td>
</tr>
<tr>
<td>UIF</td>
<td>COMIF</td>
<td>CC1IF</td>
<td>CC2IF</td>
<td>CC3IF</td>
<td>CC4IF</td>
<td>IERRF</td>
<td>DIF</td>
<td>IERF</td>
<td>TF</td>
<td>COMIF</td>
<td>CC4OF</td>
<td>CC3OF</td>
<td>CC2OF</td>
<td>CC1OF</td>
<td>B2IF</td>
</tr>
</tbody>
</table>

**Note**: The table shows the bits and their respective functions for the TIM1 status register (TIM1_SR).
Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **TERRF**: Transition error interrupt flag
   This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
   0: No encoder transition error has been detected.
   1: An encoder transition error has been detected

Bit 22  **IERRF**: Index error interrupt flag
   This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
   0: No index error has been detected.
   1: An index error has been detected

Bit 21  **DIRF**: Direction change interrupt flag
   This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
   0: No direction change
   1: Direction change

Bit 20  **IDXF**: Index interrupt flag
   This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
   0: No index event occurred.
   1: An index event has occurred

Bits 19:18  Reserved, must be kept at reset value.

Bit 17  **CC6IF**: Compare 6 interrupt flag
   Refer to CC1IF description
   *Note: Channel 6 can only be configured as output.*

Bit 16  **CC5IF**: Compare 5 interrupt flag
   Refer to CC1IF description
   *Note: Channel 5 can only be configured as output.*

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

Bit 13  **SBIF**: System break interrupt flag
   This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
   This flag must be reset to re-start PWM operation.
   0: No break event occurred.
   1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 12  **CC4OF**: Capture/compare 4 overcapture flag
   Refer to CC1OF description

Bit 11  **CC3OF**: Capture/compare 3 overcapture flag
   Refer to CC1OF description

Bit 10  **CC2OF**: Capture/compare 2 overcapture flag
   Refer to CC1OF description
Bit 9 **CC1OF**: Capture/compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set.

Bit 8 **B2IF**: Break 2 interrupt flag
This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.
0: No break event occurred.
1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred.
1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag
This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.
0: No COM event occurred.
1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag
Refer to CC1IF description.

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag
Refer to CC1IF description.
Bit 2  **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1  **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0  **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
  -- At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.
  -- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
  -- When CNT is reinitialized by a trigger event (refer to Section 41.6.3: TIM1 slave mode control register (TIM1_SMCR)), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

### 41.6.6  **TIM1 event generation register (TIM1_EGR)**

Address offset: 0x014

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B2G</td>
<td>BG</td>
<td>TG</td>
<td>COMG</td>
<td>CC4G</td>
<td>CC3G</td>
<td>CC2G</td>
<td>CC1G</td>
<td>UG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:9  Reserved, must be kept at reset value.

Bit 8  **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7  **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.
Bit 6 **TG**: Trigger generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation  
This bit can be set by software, it is automatically cleared by hardware  
0: No action  
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)  
*Note*: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation  
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation  
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation  
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: A capture/compare event is generated on channel 1:  
*If channel CC1 is configured as output:*  
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.  
*If channel CC1 is configured as input:*  
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

### 41.6.7 TIM1 capture/compare mode register 1 (TIM1_CCMR1)

Address offset: 0x018  
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).
### Input capture mode:

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:12  **IC2F[3:0]**: Input capture 2 filter

Bits 11:10  **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8  **CC2S[1:0]**: Capture/compare 2 selection

- This bitfield defines the direction of the channel (input/output) as well as the used input.
- 00: CC2 channel is configured as output
- 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
- 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
- 11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*
Bits 7:4  **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_t1 input and the length of the digital filter applied to tim_t1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING = fTIm_ker_ck, N = 2
- 0010: fSAMPLING = fTIm_ker_ck, N = 4
- 0011: fSAMPLING = fTIm_ker_ck, N = 8
- 0100: fSAMPLING = fDTS/2, N = 6
- 0101: fSAMPLING = fDTS/2, N = 8
- 0110: fSAMPLING = fDTS/4, N = 6
- 0111: fSAMPLING = fDTS/4, N = 8
- 1000: fSAMPLING = fDTS/8, N = 6
- 1001: fSAMPLING = fDTS/8, N = 8
- 1010: fSAMPLING = fDTS/16, N = 6
- 1011: fSAMPLING = fDTS/16, N = 8
- 1100: fSAMPLING = fDTS/32, N = 6
- 1101: fSAMPLING = fDTS/32, N = 8
- 1110: fSAMPLING = fDTS/32, N = 8
- 1111: fSAMPLING = fDTS/32, N = 8

Bits 3:2  **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0  **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_t1
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_t2
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

### 41.6.8 TIM1 capture/compare mode register 1 [alternate]

**TIM1_CCMR1**

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).
Output compare mode:

Bits 31:25  Reserved, must be kept at reset value.

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

Bit 15  **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12  **OC2M[3:0]**: Output compare 2 mode

Bit 11  **OC2PE**: Output compare 2 preload enable

Bit 10  **OC2FE**: Output compare 2 fast enable

Bits 9:8  **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

**Note**: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7  **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal
(tim_ocref_clr input or tim_etrf input)
Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal \( \text{tim}_\text{oc1ref} \) from which \( \text{tim}_\text{oc1} \) and \( \text{tim}_\text{oc1n} \) are derived. \( \text{tim}_\text{oc1ref} \) is active high whereas \( \text{tim}_\text{oc1} \) and \( \text{tim}_\text{oc1n} \) active level depends on CC1P and CC1NP bits.

- 0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.
- 0001: Set channel 1 to active level on match. \( \text{tim}_\text{oc1ref} \) signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
- 0010: Set channel 1 to inactive level on match. \( \text{tim}_\text{oc1ref} \) signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).
- 0011: Toggle - \( \text{tim}_\text{oc1ref} \) toggles when TIMx_CNT = TIMx_CCR1.
- 0100: Force inactive level - \( \text{tim}_\text{oc1ref} \) is forced low.
- 0101: Force active level - \( \text{tim}_\text{oc1ref} \) is forced high.
- 0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (\( \text{tim}_\text{oc1ref} = 0 \)) as long as TIMx_CNT>TIMx_CCR1 else active (\( \text{tim}_\text{oc1ref} = 1 \)).
- 0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.
- 1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on \( \text{tim}_\text{trgi} \) signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on \( \text{tim}_\text{trgi} \) signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.
- 1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on \( \text{tim}_\text{trgi} \) signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on \( \text{tim}_\text{trgi} \) signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.
- 1010: Reserved,
- 1011: Reserved,
- 1100: Combined PWM mode 1 - \( \text{tim}_\text{oc1refc} \) has the same behavior as in PWM mode 1. \( \text{tim}_\text{oc1refc} \) is the logical OR between \( \text{tim}_\text{oc1ref} \) and \( \text{tim}_\text{oc2ref} \).
- 1101: Combined PWM mode 2 - \( \text{tim}_\text{oc1refc} \) has the same behavior as in PWM mode 2. \( \text{tim}_\text{oc1refc} \) is the logical AND between \( \text{tim}_\text{oc1ref} \) and \( \text{tim}_\text{oc2ref} \).
- 1110: Asymmetric PWM mode 1 - \( \text{tim}_\text{oc1refc} \) has the same behavior as in PWM mode 1. \( \text{tim}_\text{oc1refc} \) outputs \( \text{tim}_\text{oc1ref} \) when the counter is counting up, \( \text{tim}_\text{oc2ref} \) when it is counting down.
- 1111: Asymmetric PWM mode 2 - \( \text{tim}_\text{oc1refc} \) has the same behavior as in PWM mode 2. \( \text{tim}_\text{oc1ref} \) outputs \( \text{tim}_\text{oc1ref} \) when the counter is counting up, \( \text{tim}_\text{oc2ref} \) when it is counting down.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

**Note:** In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

**Note:** On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
### TIM1 capture/compare mode register 2 (TIM1_CCMR2)

**Address offset:** 0x001C  
**Reset value:** 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

#### Input capture mode

<table>
<thead>
<tr>
<th>Bit 32</th>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Input capture mode**

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:12: **IC4F[3:0]:** Input capture 4 filter
- Bits 11:10: **IC4PSC[1:0]:** Input capture 4 prescaler
Advanced-control timers (TIM1) RM0477

41.6.10 TIM1 capture/compare mode register 2 [alternate] (TIM1_CCMR2)

Address offset: 0x01C
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

Bits 9:8 CC4S[1:0]: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 IC3F[3:0]: Input capture 3 filter

Bits 3:2 IC3PSC[1:0]: Input capture 3 prescaler

Bits 1:0 CC3S[1:0]: Capture/compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.
Bits 23:17 Reserved, must be kept at reset value.
Bit 15 OC4CE: Output compare 4 clear enable

Bits 24, 14:12 OC4M[3:0]: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 OC4PE: Output compare 4 preload enable

Bit 10 OC4FE: Output compare 4 fast enable
Bits 9:8  **CC4S[1:0]**: Capture/compare 4 selection

- 00: CC4 channel is configured as output
- 01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
- 10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
- 11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note*: **CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).**

Bit 7  **OC3CE**: Output compare 3 clear enable
Bits 16, 6:4  **OC3M[3:0]: Output compare 3 mode**

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs. (this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

*Note:* These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

*Note:* In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.
Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

- This bitfield defines the direction of the channel (input/output) as well as the used input.
  - 00: CC3 channel is configured as output
  - 01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
  - 10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
  - 11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).*

### 41.6.11 TIM1 capture/compare enable register (TIM1_CCER)

Address offset: 0x020

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC4NP</td>
<td>CC4NE</td>
<td>CC4P</td>
<td>CC4E</td>
<td>CC3NP</td>
<td>CC3NE</td>
<td>CC3P</td>
<td>CC3E</td>
<td>CC2NP</td>
<td>CC2NE</td>
<td>CC2P</td>
<td>CC2E</td>
<td>CC1NP</td>
<td>CC1NE</td>
<td>CC1P</td>
<td>CC1E</td>
</tr>
</tbody>
</table>

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

- Bit 21 **CC6P**: Capture/compare 6 output polarity
  
  Refer to CC1P description

- Bit 20 **CC6E**: Capture/compare 6 output enable
  
  Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

- Bit 17 **CC5P**: Capture/compare 5 output polarity
  
  Refer to CC1P description

- Bit 16 **CC5E**: Capture/compare 5 output enable
  
  Refer to CC1E description

- Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
  
  Refer to CC1NP description

- Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
  
  Refer to CC1NE description

- Bit 13 **CC4P**: Capture/compare 4 output polarity
  
  Refer to CC1P description

- Bit 12 **CC4E**: Capture/compare 4 output enable
  
  Refer to CC1E description

- Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
  
  Refer to CC1NP description
Bit 10  **CC3NE**: Capture/compare 3 complementary output enable  
Refer to CC1NE description

Bit 9  **CC3P**: Capture/compare 3 output polarity  
Refer to CC1P description

Bit 8  **CC3E**: Capture/compare 3 output enable  
Refer to CC1E description

Bit 7  **CC2NP**: Capture/compare 2 complementary output polarity  
Refer to CC1NP description

Bit 6  **CC2NE**: Capture/compare 2 complementary output enable  
Refer to CC1NE description

Bit 5  **CC2P**: Capture/compare 2 output polarity  
Refer to CC1P description

Bit 4  **CC2E**: Capture/compare 2 output enable  
Refer to CC1E description

Bit 3  **CC1NP**: Capture/compare 1 complementary output polarity  

**CC1 channel configured as output:**  
0: tim_oc1n active high.  
1: tim_oc1n active low.

**CC1 channel configured as input:**  
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.  
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2  **CC1NE**: Capture/compare 1 complementary output enable  
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.  
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.
Bit 1  **CC1P**: Capture/compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of T11FP1 and T12FP1 for trigger or capture operations.
CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to T1xFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode or encoder mode).
CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to T1xFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is inverted (trigger operation in gated mode or encoder mode).
CC1NP = 1, CC1P = 1: non-inverted/both edges. The circuit is sensitive to both T1xFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
CC1NP = 1, CC1P = 0: the configuration is reserved, it must not be used.

**Note**: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

**Note**: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0  **CC1E**: Capture/compare 1 output enable
0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to Table 422 for details.

**Note**: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.
Table 422. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.
41.6.12 TIM1 counter (TIM1_CNT)

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 30:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>CNT[15:0]: Counter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-dithering mode (DITHEN = 0)</td>
<td></td>
</tr>
<tr>
<td>The register holds the counter value.</td>
<td></td>
</tr>
</tbody>
</table>

Dithering mode (DITHEN = 1)

| The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available. |

41.6.13 TIM1 prescaler (TIM1_PSC)

Address offset: 0x028
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The counter clock frequency ( f_{\text{tim_cnt_ck}} ) is equal to ( f_{\text{tim_psc_ck}} / (\text{PSC[15:0]} + 1) ).</td>
<td></td>
</tr>
</tbody>
</table>

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).
41.6.14  TIM1 autoreload register (TIM1_ARR)

Address offset: 0x02C
Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>Bits 31:20</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 19:0</td>
<td>ARR[19:0]: Autoreload value</td>
</tr>
<tr>
<td></td>
<td>ARR is the value to be loaded in the actual autoreload register.</td>
</tr>
<tr>
<td></td>
<td>Refer to the Section 41.3.3: Time-base unit for more details about ARR update and behavior.</td>
</tr>
<tr>
<td></td>
<td>The counter is blocked while the autoreload value is null.</td>
</tr>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the autoreload value.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.</td>
</tr>
</tbody>
</table>

41.6.15  TIM1 repetition counter register (TIM1_RCR)

Address offset: 0x030
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>REP[15:0]: Repetition counter reload value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.</td>
</tr>
<tr>
<td></td>
<td>When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.</td>
</tr>
<tr>
<td></td>
<td>It means in PWM mode (REP+1) corresponds to:</td>
</tr>
<tr>
<td></td>
<td>– the number of PWM periods in edge-aligned mode</td>
</tr>
<tr>
<td></td>
<td>– the number of half PWM period in center-aligned mode.</td>
</tr>
</tbody>
</table>
41.6.16 TIM1 capture/compare register 1 (TIM1_CCR1)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR1[19:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

41.6.17 TIM1 capture/compare register 2 (TIM1_CCR2)

Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR2[19:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR2[19:0]**: Capture/compare 2 value

*If channel CC2 is configured as output*: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

*If channel CC2 is configured as input*: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

### 41.6.18  TIM1 capture/compare register 3 (TIM1_CCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**CCR3[19:16]**

**CCR3[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR3[19:0]: Capture/compare value**

*If channel CC3 is configured as output*: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on TIM_OC3 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.

*If channel CC3 is configured as input*: CCR3 is the counter value transferred by the last input capture 3 event (TIM_IC3). The TIMx_CCR3 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

### 41.6.19  TIM1 capture/compare register 4 (TIM1_CCR4)

**Address offset**: 0x040

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**CCR4[15:16]**

**CCR4[15:0]**

**reset value**: 0x0000 0000

**Address offset**: 0x040

**Register**: TIM1_CCR4

**Description**: Capture/compare register 4 for TIM1

**Access**: Read/write

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR4[19:0]: Capture/compare value**

**If channel CC4 is configured as output:** CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc4 output.

*Non-dithering mode (DITHEN = 0)*

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

*Dithering mode (DITHEN = 1)*

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

**If channel CC4 is configured as input:** CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

*Non-dithering mode (DITHEN = 0)*

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

*Dithering mode (DITHEN = 1)*

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

### 41.6.20  TIM1 break and dead-time register (TIM1_BDTR)

Address offset: 0x044

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
</tr>
<tr>
<td>29</td>
<td>BK2BID</td>
</tr>
<tr>
<td>28</td>
<td>BKBID</td>
</tr>
<tr>
<td>27</td>
<td>BK2DSRM</td>
</tr>
<tr>
<td>26</td>
<td>BKDSRM</td>
</tr>
<tr>
<td>25</td>
<td>BK2P</td>
</tr>
<tr>
<td>24</td>
<td>BK2E</td>
</tr>
<tr>
<td>23</td>
<td>BK2F[3:0]</td>
</tr>
<tr>
<td>22</td>
<td>BKF[3:0]</td>
</tr>
<tr>
<td>21</td>
<td>MOE</td>
</tr>
<tr>
<td>20</td>
<td>AOE</td>
</tr>
<tr>
<td>19</td>
<td>BKP</td>
</tr>
<tr>
<td>18</td>
<td>BKE</td>
</tr>
<tr>
<td>17</td>
<td>OSSR</td>
</tr>
<tr>
<td>16</td>
<td>OSSI</td>
</tr>
<tr>
<td>15</td>
<td>LOCK[1:0]</td>
</tr>
<tr>
<td>14</td>
<td>DTG[7:0]</td>
</tr>
</tbody>
</table>

**Note:** As the bits BK2BID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSR, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.
Bits 31:30  Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBDID description

Bit 28 **BKBDID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBDID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high

*Note:* This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection (including all sources connected to bk_acth and BKIN sources, as per Figure 444: Break and Break2 circuitry overview).
0: Break2 function disabled
1: Break2 function enabled

*Note:* The BKIN2 must only be used with OSSR = OSSI = 1.

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), \( N = 2 \)
0010: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), \( N = 4 \)
0011: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), \( N = 8 \)
0100: \( f_{\text{SAMPLING}} = f_{DTS}/2 \), \( N = 6 \)
0101: \( f_{\text{SAMPLING}} = f_{DTS}/2 \), \( N = 8 \)
0110: \( f_{\text{SAMPLING}} = f_{DTS}/4 \), \( N = 6 \)
0111: \( f_{\text{SAMPLING}} = f_{DTS}/4 \), \( N = 8 \)
1000: \( f_{\text{SAMPLING}} = f_{DTS}/6 \), \( N = 6 \)
1001: \( f_{\text{SAMPLING}} = f_{DTS}/6 \), \( N = 8 \)
1010: \( f_{\text{SAMPLING}} = f_{DTS}/16 \), \( N = 5 \)
1011: \( f_{\text{SAMPLING}} = f_{DTS}/16 \), \( N = 6 \)
1100: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 5 \)
1101: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 6 \)
1110: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 8 \)
1111: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 8 \)

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), \( N = 2 \)
0010: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), \( N = 4 \)
0011: \( f_{\text{SAMPLING}} = f_{\text{tim_ker_ck}} \), \( N = 8 \)
0100: \( f_{\text{SAMPLING}} = f_{DTS}/2 \), \( N = 6 \)
0101: \( f_{\text{SAMPLING}} = f_{DTS}/2 \), \( N = 8 \)
0110: \( f_{\text{SAMPLING}} = f_{DTS}/4 \), \( N = 6 \)
0111: \( f_{\text{SAMPLING}} = f_{DTS}/4 \), \( N = 8 \)
1000: \( f_{\text{SAMPLING}} = f_{DTS}/6 \), \( N = 6 \)
1001: \( f_{\text{SAMPLING}} = f_{DTS}/6 \), \( N = 8 \)
1010: \( f_{\text{SAMPLING}} = f_{DTS}/16 \), \( N = 5 \)
1011: \( f_{\text{SAMPLING}} = f_{DTS}/16 \), \( N = 6 \)
1100: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 5 \)
1101: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 6 \)
1110: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 8 \)
1111: \( f_{\text{SAMPLING}} = f_{DTS}/32 \), \( N = 8 \)

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled

In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).

See OC/OCN enable description for more details ([Section 41.6.11: TIM1 capture/compare enable register (TIM1_CCER)]).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to bk_acth and BKIN sources, as per [Figure 444: Break and Break2 circuitry overview]).

0: Break function disabled

1: Break function enabled

*Note:* This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details ([Section 41.6.11: TIM1 capture/compare enable register (TIM1_CCER)]).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

*Note:* This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.

See OC/OCN enable description for more details (Section 41.6.11: TIM1 capture/compare enable register (TIM1_CCER)).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

*Note*: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBI/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note*: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx => DT = DTG[7:0]x t_dtg with t_dtg = tDTS.

DTG[7:5] = 10x => DT = (64+DTG[5:0])xtdtg with Tdtg = 2xTdts.

DTG[7:5] = 110 => DT = (32+DTG[5:0])xtdtg with Tdtg = 8xTdts.

DTG[7:5] = 111 => DT = (32+DTG[5:0])xtdtg with Tdtg = 16xTdts.

Example if T_DTS = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

*Note*: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

### 41.6.21 TIM1 capture/compare register 5 (TIM1_CCR5)

**Address offset**: 0x048

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CCR5[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

1754/3791  RM0477 Rev 6
Advanced-control timers (TIM1)

41.6.22 TIM1 capture/compare register 6 (TIM1_CCR6)

Address offset: 0x04C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>GC5C3</th>
<th>Group channel 5 and channel 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion on channel 3 output:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: No effect of tim_oc5ref on tim_oc3refc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: it is also possible to apply this distortion on combined PWM signals.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>GC5C2</th>
<th>Group channel 5 and channel 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion on channel 2 output:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: No effect of tim_oc5ref on tim_oc2refc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: it is also possible to apply this distortion on combined PWM signals.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>GC5C1</th>
<th>Group channel 5 and channel 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion on channel 1 output:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: No effect of oc5ref on oc1refc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: oc1refc is the logical AND of oc1ref and oc5ref</td>
<td></td>
<td></td>
</tr>
<tr>
<td>This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: it is also possible to apply this distortion on combined PWM signals.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 CCR5[19:0]: Capture/compare 5 value

CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.

Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.

Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.
41.6.23 TIM1 capture/compare mode register 3 (TIM1_CCMR3)

Address offset: 0x050
Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

| Bits 31:25 | Reserved, must be kept at reset value. |
| Bit 23:17 | Reserved, must be kept at reset value. |
| Bit 15 | OC6CE: Output compare 6 clear enable |
| Bits 24, 14:12 | OC6M[3:0]: Output compare 6 mode |
| Bit 11 | OC6PE: Output compare 6 preload enable |
| Bit 10 | OC6FE: Output compare 6 fast enable |
| Bits 9:8 | Reserved, must be kept at reset value. |
| Bit 7 | OC5CE: Output compare 5 clear enable |
| Bits 16, 6:4 | OC5M[3:0]: Output compare 5 mode |
| Bit 3 | OC5PE: Output compare 5 preload enable |
| Bit 2 | OC5FE: Output compare 5 fast enable |
| Bits 1:0 | Reserved, must be kept at reset value. |
### 41.6.24 TIM1 timer deadtime register 2 (TIM1_DTR2)

**Address offset:** 0x054  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

**Bit 17** **DTPE**: Deadtime preload enable  
0: Deadtime value is not preloaded  
1: Deadtime value preload is enabled  
*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

**Bit 16** **DTAE**: Deadtime asymmetric enable  
0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register  
1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.  
*Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

**Bits 15:8** Reserved, must be kept at reset value.

**Bits 7:0** **DTGF[7:0]**: Dead-time falling edge generator setup  
This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.  
DTGF[7:5] = 0xx => DTF = DTGF[7:0] \* tdtg with tdtg = tDTS.  
DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])\*xtdtg with Tdtg = 2\*tDTS.  
DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])\*xtdtg with Tdtg = 4\*tDTS.  
DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])\*xtdtg with Tdtg = 8\*tDTS.  
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:  
0 to 15875 ns by 125 ns steps,  
16 μs to 31750 ns by 250 ns steps,  
32 μs to 63 μs by 1 μs steps,  
64 μs to 126 μs by 2 μs steps  
*Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).
### 41.6.25 TIM1 timer encoder control register (TIM1_ECR)

Address offset: 0x058  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

**Bits 26:24**  
**PWPRSC[2:0]:** Pulse width prescaler  
This bitfield sets the clock prescaler for the pulse generator, as following:  
\[ t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim.ker.ck} \]

**Bits 23:16**  
**PW[7:0]:** Pulse width  
This bitfield defines the pulse duration, as following:  
\[ t_{PW} = PW[7:0] \times t_{PWG} \]

**Bits 15:8**  
Reserved, must be kept at reset value.

**Bits 7:6**  
**IPOS[1:0]:** Index positioning  
In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.  
00: Index resets the counter when AB = 00  
01: Index resets the counter when AB = 01  
10: Index resets the counter when AB = 10  
11: Index resets the counter when AB = 11  
In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.  
x0: Index resets the counter when clock is 0  
x1: Index resets the counter when clock is 1  
*Note:* **IPOS[1]** bit is not significant

**Bit 5**  
**FIDX:** First index  
This bit indicates if the first index only is taken into account  
0: Index is always active  
1: the first Index only resets the counter
41.6.26  TIM1 timer input selection register (TIM1_TISEL)

Address offset: 0x05C
Reset value: 0x0000 0000

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 TI4SEL[3:0]: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
... 
1111: tim_ti4_in15
Refer to Section 41.3.2: TIM1 pins and internal signals for interconnects list.

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

Bits 19:16 TI3SEL[3:0]: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
... 
1111: tim_ti3_in15
Refer to Section 41.3.2: TIM1 pins and internal signals for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.
Advanced-control timers (TIM1) RM0477

41.6.27 TIM1 alternate function option register 1 (TIM1_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETRSEL[3:2]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 17:14 **ETRSEL[3:0]: etr_in source selection**

These bits select the etr_in input source.

0000: tim_etr0: TIMx_ETR input

0001: tim_etr1

... 1111: tim_etr15

Refer to Section 41.3.2: TIM1 pins and internal signals for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P: tim_brk_cmp4 input polarity**

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 12 **BKCMP3P**: tim\_brk\_cmp3 input polarity

This bit selects the tim\_brk\_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim\_brk\_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim\_brk\_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 11 **BKCMP2P**: tim\_brk\_cmp2 input polarity

This bit selects the tim\_brk\_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim\_brk\_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim\_brk\_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 10 **BKCMP1P**: tim\_brk\_cmp1 input polarity

This bit selects the tim\_brk\_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim\_brk\_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim\_brk\_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 9 **BKINP**: TIMx\_BKin input polarity

This bit selects the TIMx\_BKin alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx\_BKin input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: TIMx\_BKin input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 8 **BKCMP8E**: tim\_brk\_cmp8 enable

This bit enables the tim\_brk\_cmp8 for the timer’s tim\_brk input. tim\_brk\_cmp8 output is 'ORed' with the other tim\_brk sources.

0: tim\_brk\_cmp8 input disabled
1: tim\_brk\_cmp8 input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 7 **BKCMP7E**: tim\_brk\_cmp7 enable

This bit enables the tim\_brk\_cmp7 for the timer’s tim\_brk input. tim\_brk\_cmp7 output is 'ORed' with the other tim\_brk sources.

0: tim\_brk\_cmp7 input disabled
1: tim\_brk\_cmp7 input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).

Bit 6 **BKCMP6E**: tim\_brk\_cmp6 enable

This bit enables the tim\_brk\_cmp6 for the timer’s tim\_brk input. tim\_brk\_cmp6 output is 'ORed' with the other tim\_brk sources.

0: tim\_brk\_cmp6 input disabled
1: tim\_brk\_cmp6 input enabled

*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx\_BDTR register).
Bit 5 **BKCMP5E**: tim_brk_cmp5 enable
This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp5 input disabled
1: tim_brk_cmp5 input enabled
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable
This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp4 input disabled
1: tim_brk_cmp4 input enabled
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable
This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp3 input disabled
1: tim_brk_cmp3 input enabled
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable
This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp2 input disabled
1: tim_brk_cmp2 input enabled
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable
This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.
0: tim_brk_cmp1 input disabled
1: tim_brk_cmp1 input enabled
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable
This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.
0: TIMx_BKIN input disabled
1: TIMx_BKIN input enabled
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Refer to **Section 41.3.2: TIM1 pins and internal signals** for product specific implementation.
41.6.28 TIM1 alternate function register 2 (TIM1_AF2)

Address offset: 0x064
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bits 31:19</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 18:16</td>
<td>OCRSEL[2:0]: ocref_clr source selection</td>
</tr>
<tr>
<td></td>
<td>These bits select the ocref_clr input source.</td>
</tr>
<tr>
<td></td>
<td>000: tim_ocref_clr0</td>
</tr>
<tr>
<td></td>
<td>001: tim_ocref_clr1</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>111: tim_ocref_clr7</td>
</tr>
<tr>
<td></td>
<td>Refer to Section 41.3.2: TIM1 pins and internal signals for product specific information.</td>
</tr>
<tr>
<td>Note:</td>
<td>These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).</td>
</tr>
</tbody>
</table>

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

Bit 13 BK2CMP4P: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 BK2CMP3P: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 BK2CMP2P: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity
This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.
0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity
This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.
0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)
1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable
This bit enables the tim_brk2_cmp8 for the timer’s tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.
0: tim_brk2_cmp8 input disabled
1: tim_brk2_cmp8 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable
This bit enables the tim_brk2_cmp7 for the timer’s tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.
0: tim_brk2_cmp7 input disabled
1: tim_brk2_cmp7 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable
This bit enables the tim_brk2_cmp6 for the timer’s tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.
0: tim_brk2_cmp6 input disabled
1: tim_brk2_cmp6 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable
This bit enables the tim_brk2_cmp5 for the timer’s tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.
0: tim_brk2_cmp5 input disabled
1: tim_brk2_cmp5 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable
This bit enables the tim_brk2_cmp4 for the timer’s tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.
0: tim_brk2_cmp4 input disabled
1: tim_brk2_cmp4 input enabled
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
### 41.6.29 TIM1 DMA control register (TIM1_DCR)

Address offset: 0x3DC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 3 BK2CMP3E:** tim_brk2_cmp3 enable  
This bit enables the tim_brk2_cmp3 for the timer’s tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.  
0: tim_brk2_cmp3 input disabled  
1: tim_brk2_cmp3 input enabled  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

**Bit 2 BK2CMP2E:** tim_brk2_cmp2 enable  
This bit enables the tim_brk2_cmp2 for the timer’s tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.  
0: tim_brk2_cmp2 input disabled  
1: tim_brk2_cmp2 input enabled  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

**Bit 1 BK2CMP1E:** tim_brk2_cmp1 enable  
This bit enables the tim_brk2_cmp1 for the timer’s tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.  
0: tim_brk2_cmp1 input disabled  
1: tim_brk2_cmp1 input enabled  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

**Bit 0 BK2INE:** TIMx_BKIN2 input enable  
This bit enables the TIMx_BKIN2 alternate function input for the timer’s tim_brk2 input. TIMx_BKIN2 input is ‘ORed’ with the other tim_brk2 sources.  
0: TIMx_BKIN2 input disabled  
1: TIMx_BKIN2 input enabled  
*Note:* This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

*Note:* Refer to Section 41.3.2: TIM1 pins and internal signals for product specific implementation.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  **DBSS[3:0]: DMA burst source selection**

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

- 0000: Reserved
- 0001: Update
- 0010: CC1
- 0011: CC2
- 0100: CC3
- 0101: CC4
- 0110: COM
- 0111: Trigger
- Others: reserved

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]: DMA burst length**

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e., the number of transfers. Transfers can be in half-words or in bytes (see example below).

- 00000: 1 transfer
- 00001: 2 transfers
- 00010: 3 transfers
- ...
- 11010: 26 transfers

**Example:** Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

- If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:
  
  \[(\text{TIMx_CR1 address}) + \text{DBA} + \text{(DMA index)}\]

  In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

  According to the configuration of the DMA Data Size, several cases may occur:

  - If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.
  - If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on.

So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **DBA[4:0]: DMA base address**

This 5-bit vector defines the base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

**Example:**

- 00000: TIMx_CR1
- 00001: TIMx_CR2
- 00010: TIMx_SMCR
- ...
41.6.30 TIM1 DMA address for full transfer (TIM1_DMAR)

Address offset: 0x3E0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 DMAB[31:0]: DMA register for burst accesses
A read or write operation to the DMAR register accesses the register located at the address
(TIMx_CR1 address) + (DBA + DMA index) x 4
where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base
address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA
transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

41.6.31 TIM1 register map

TIM1 registers are mapped as 16-bit addressable registers as described in the table below:

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>TIMx_SMCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td></td>
</tr>
</tbody>
</table>
### Table 423. TIM1 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Address</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x018</td>
<td>TIMx_CCMR1</td>
<td>IC2F</td>
<td>IC2 PSC [1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC2</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IC1F</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC1</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC2M</td>
<td>[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC2PE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC2S</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC2SE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC2SE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC1M</td>
<td>[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC1C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC1CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC1F</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC1FE</td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>TIMx_CCMR1</td>
<td>IC4F</td>
<td>IC4 PSC [1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC4</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IC3F</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC3</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC4M</td>
<td>[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC4PE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC4S</td>
<td>[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC4SE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC4SE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC3M</td>
<td>[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC3C</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CC3CE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC3P</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>OC3PE</td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>TIMx_CCMR2</td>
<td>CCR1</td>
<td>CCR1[19:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCR2</td>
<td>CCR2[19:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCR3</td>
<td>CCR3[19:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CCR4</td>
<td>CCR4[19:0]</td>
</tr>
<tr>
<td>0x024</td>
<td>TIMx_CNT</td>
<td>UFGP1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CNT</td>
<td>CNT[15:0]</td>
</tr>
<tr>
<td>0x028</td>
<td>TIMx_PSC</td>
<td>PSC</td>
<td>PSC[15:0]</td>
</tr>
<tr>
<td>0x02C</td>
<td>TIMx_ARR</td>
<td>REP</td>
<td>REP[15:0]</td>
</tr>
<tr>
<td>0x030</td>
<td>TIMx_RCR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>TIMx_CCR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x038</td>
<td>TIMx_CCR2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03C</td>
<td>TIMx_CCR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>TIMx_CCR4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x044</td>
<td>TIMx_BDTR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reset value**

- TIMx_CCMR1
  - Reset value: 0
- TIMx_CCMR2
  - Reset value: 0
- TIMx_PSC
  - Reset value: 0
- TIMx_ARR
  - Reset value: 0
- TIMx_RCR
  - Reset value: 0
- TIMx_CCR1
  - Reset value: 0
- TIMx_CCR2
  - Reset value: 0
- TIMx_CCR3
  - Reset value: 0
- TIMx_CCR4
  - Reset value: 0
- TIMx_BDTR
  - Reset value: 0
### Table 423. TIM1 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x048</td>
<td>TIMx_CCR5</td>
<td>0x04C</td>
<td>TIMx_CCR6</td>
<td>0x050</td>
<td>TIMx_CCMR3</td>
<td>0x054</td>
<td>TIMx_DTR2</td>
</tr>
<tr>
<td>0x058</td>
<td>TIMx_ECR</td>
<td>0x05C</td>
<td>TIMx_TISel</td>
<td>0x060</td>
<td>TIMx_AF1</td>
<td>0x064</td>
<td>TIMx_AF2</td>
</tr>
<tr>
<td>0x068..</td>
<td>Reserved</td>
<td>0x06C</td>
<td>TIMx_DCR</td>
<td>0x070..</td>
<td>Reserved</td>
<td>0x074</td>
<td>TIMx_DMAR</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
42 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

42.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare and PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 42.4.23: Timer synchronization.

42.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

• 16-bit or 32-bit up, down, up/down autoreload counter.
• 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
• Up to four independent channels for:
  – Input capture.
  – Output compare.
  – PWM generation (edge- and center-aligned modes).
  – One-pulse mode output.
• Synchronization circuit to control the timer with external signals and to interconnect several timers.
• Interrupt/DMA generation on the following events:
  – Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  – Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  – Input capture.
  – Output compare.
• Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
• Trigger input for external clock or cycle-by-cycle current management.
42.3 TIM2/TIM3/TIM4/TIM5 implementation

Table 424. STM32H7Rx/7Sx general purpose timers

<table>
<thead>
<tr>
<th>Timer instance</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>32-bit</td>
<td>32-bit</td>
<td>32-bit</td>
<td>32-bit</td>
</tr>
<tr>
<td>OCREF clear selection Sources</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Sources</td>
<td>tim_etrf</td>
<td>tim_etrf</td>
<td>tim_etrf</td>
<td>tim_etrf</td>
</tr>
</tbody>
</table>
42.4 TIM2/TIM3/TIM4/TIM5 functional description

42.4.1 Block diagram

Figure 486. General-purpose timer block diagram

1. This feature is not available on all timers, refer to Section 42.3: TIM2/TIM3/TIM4/TIM5 implementation.
42.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 425 and Table 426 in this section summarize the TIM inputs and outputs.

Table 425. TIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.</td>
</tr>
<tr>
<td>TIM_CH2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_ETR</td>
<td>Input</td>
<td>External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etrt_in prescaler is used.</td>
</tr>
</tbody>
</table>

Table 426. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.</td>
</tr>
<tr>
<td>tim_ti2_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti3_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_ti4_in[15:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_etrt[15:0]</td>
<td>Input</td>
<td>External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etrt_in prescaler is used.</td>
</tr>
<tr>
<td>tim_itr[15:0]</td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_trgo</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_ocref_clr[7:0]</td>
<td>Input</td>
<td>Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocref_clr signals, typically for hardware cycle-by-cycle pulse width control.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock.</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock</td>
</tr>
</tbody>
</table>
### Table 426. TIM internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_it</td>
<td>Output</td>
<td>Global Timer interrupt, gathering capture/compare, update and break trigger requests.</td>
</tr>
<tr>
<td>tim_upd_dma</td>
<td>Output</td>
<td>Timer update dma request.</td>
</tr>
<tr>
<td>tim_trgi_dma</td>
<td>Output</td>
<td>Timer trigger dma request.</td>
</tr>
</tbody>
</table>

*Table 427, Table 428, Table 429 and Table 430 are listing the sources connected to the tim_ti[4:1] input multiplexers.*

### Table 427. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in0</td>
<td>TIM2_CH1</td>
<td>TIM3_CH1</td>
<td>TIM4_CH1</td>
<td>TIM5_CH1</td>
</tr>
<tr>
<td>tim_ti1_in1</td>
<td>eth_ptp_pps_out</td>
<td>eth_ptp_pps_out</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_ti1_in[15:2]</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 428. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti2_in0</td>
<td>TIM2_CH2</td>
<td>TIM3_CH2</td>
<td>TIM4_CH2</td>
<td>TIM5_CH2</td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 429. Interconnect to the tim_ti3 input multiplexer

<table>
<thead>
<tr>
<th>Source</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti3_in0</td>
<td>TIM2_CH3</td>
<td>TIM3_CH3</td>
<td>TIM4_CH3</td>
<td>TIM5_CH3</td>
</tr>
<tr>
<td>tim_ti3_in[15:1]</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 430. Interconnect to the tim_ti4 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti4 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti4_in0</td>
<td>TIM2_CH4</td>
</tr>
<tr>
<td></td>
<td>TIM3_CH4</td>
</tr>
<tr>
<td></td>
<td>TIM4_CH4</td>
</tr>
<tr>
<td></td>
<td>TIM5_CH4</td>
</tr>
<tr>
<td>tim_ti4_in[15:1]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 431 lists the internal sources connected to the tim_etr input multiplexer.

Table 431. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>TIMx</th>
<th>TIM2</th>
<th>TIM3</th>
<th>TIM4</th>
<th>TIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>Reserved</td>
<td>tim2_trgo</td>
<td>tim2_trgo</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
<td>Reserved</td>
<td>tim3_trgo</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
<td>tim4_trgo</td>
<td>Reserved</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>tim_itr4</td>
<td>tim5_trgo</td>
<td>tim5_trgo</td>
<td>tim5_trgo</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_itr5</td>
<td>tim9_trgo</td>
<td>tim9_trgo</td>
<td>tim9_trgo</td>
<td>tim9_trgo</td>
</tr>
<tr>
<td>tim_itr6</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>tim_itr7</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
<td>tim13_oc1</td>
</tr>
<tr>
<td>tim_itr8</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
<td>tim14_oc1</td>
</tr>
<tr>
<td>tim_itr9</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
<td>tim15_trgo</td>
</tr>
<tr>
<td>tim_itr10</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
<td>tim16_oc1</td>
</tr>
<tr>
<td>tim_itr11</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
<td>tim17_oc1</td>
</tr>
<tr>
<td>tim_itr12</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_itr13</td>
<td>OTG_HS_SOF</td>
<td>Reserved</td>
<td></td>
<td>OTG_HS_SOF</td>
</tr>
<tr>
<td>tim_itr14</td>
<td>OTG_FS_SOF</td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_itr15</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 432 lists the internal sources connected to the tim_etr input multiplexer.

Table 432. Interconnect to the tim_etr input multiplexer

<table>
<thead>
<tr>
<th>Timer external trigger input signal</th>
<th>Timer external trigger signals assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_et0</td>
<td>TIM2_ETR</td>
</tr>
<tr>
<td>tim_et1</td>
<td>DCMI_HSYNC</td>
</tr>
<tr>
<td>tim_et2</td>
<td>LTDC_HSYNC</td>
</tr>
</tbody>
</table>
42.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set one clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC...
register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 487* and *Figure 488* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 487. Counter timing diagram with prescaler division change from 1 to 2**

![Counter timing diagram with prescaler division change from 1 to 2](image)
42.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.
Figure 489. Counter timing diagram, internal clock divided by 1

Figure 490. Counter timing diagram, internal clock divided by 2
Figure 491. Counter timing diagram, internal clock divided by 4

- **tim_psc_ck**: Pulses
- **CEN**: Green bar
- **tim_cnt_ck**: Pulses
- **Counter register**: 0035 → 0036 → 0000 → 0001
- **Counter overflow**: Green bar
- **Update event (UEV)**: Green bar
- **Update interrupt flag (UIF)**: Green bar

Figure 492. Counter timing diagram, internal clock divided by N

- **tim_psc_ck**: Pulses
- **tim_cnt_ck**: Pulses
- **Counter register**: 1F → 20 → 00
- **Counter overflow**: Green bar
- **Update event (UEV)**: Green bar
- **Update interrupt flag (UIF)**: Green bar
Figure 493. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

- `tim_psc_ck`
- `CEN`
- `tim_cnt_ck`
- `Counter register`: 31 32 33 34 35 36 00 01 02 03 04 05 06 07
- `Counter overflow`
- `Update event (UEV)`
- `Update interrupt flag (UIF)`
- `Auto-reload preload register`: FF 36

Write a new value in TIMx_ARR
Figure 494. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)

Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 495. Counter timing diagram, internal clock divided by 1
Figure 496. Counter timing diagram, internal clock divided by 2

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register**: 0002, 0001, 0000, 0036, 0035, 0034, 0033
- **Counter underflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**

Figure 497. Counter timing diagram, internal clock divided by 4

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- **Counter register**: 0001, 0000, 0000, 0001
- **Counter underflow**
- **Update event (UEV)**
- **Update interrupt flag (UIF)**
### Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the
autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.
Figure 500. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6

1. Here, center-aligned mode 1 is used (for more details refer to Section 42.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5)).

Figure 501. Counter timing diagram, internal clock divided by 2
1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

**Figure 502. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36**

- **tim_psc_ck**
- **CEN**
- **tim_cnt_ck**
- Counter register: 0034 0035 0036 0035
- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

Note: Here, center_aligned mode 2 or 3 is updated with an UIF on overflow

**Figure 503. Counter timing diagram, internal clock divided by N**

- **tim_psc_ck**
- **tim_cnt_ck**
- Counter register: 20 1F 01 00
- Counter underflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 504. Counter timing diagram, Update event with ARPE = 1 (counter underflow)
42.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_t1 or tim_t2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to Using one timer as prescaler for another timer for more details.

**Internal clock source (tim_ker_ck)**

If the slave mode controller is disabled (SMS = 000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register), and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

*Figure 506* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

For example, to configure the upcounter to count in response to a rising edge on the TIM2 input, use the following procedure:

1. Select the proper TIM2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the TIM2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 508. Control circuit in external clock mode 1

External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 509 gives an overview of the external trigger input block.
For example, to configure the upcounter to count each two rising edges on TIM_ETR_in, use the following procedure:

1. Select the proper TIM_ETR_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register.
4. Select rising edge detection on the TIM_ETR_in by writing ETP = 0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each two TIM_ETR_in rising edges.

The delay between the rising edge on TIM_ETR_in and the actual clock of the counter is due to the resynchronization circuit on the TIM_ETRP signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of TIMxCLK frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.
42.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 511. Capture/compare channel (example: channel 1 input stage)
The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

**Figure 512. Capture/compare channel 1 main circuit**

![Capture/compare channel 1 main circuit diagram]

**Figure 513. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)**

![Output stage of capture/compare channel diagram]

1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.
In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

**42.4.7 Input capture mode**

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCXIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

**Note:** IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.
42.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two T1xFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_t1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_t1 selected).
3. Select the active polarity for tim_t1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_t1 selected).
5. Select the active polarity for tim_t1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_t1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.
42.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (TIMx_OCxref and then TIMx_OCx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (TIMx_OCxref/TIMx_OCx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus TIMx_OCxref is forced high (TIMx_OCxref is always active high) and TIMx_OCx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (TIMx_OCx active high) => TIMx_OCx is forced to high level.

TIMx_OCxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare mode section.

42.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only TIMx_CH1fp1 and TIMx_CH2fp2 are connected to the slave mode controller.
active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

**Procedure**

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 515*. 
42.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxE bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_ocref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 000) to one of the PWM modes (OCxM = 110 or 111). This forces the PWM by software while the timer is running.

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.
PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to Up-counting mode.

In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. Figure 516 shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

![Figure 516. Edge-aligned PWM waveforms (ARR = 8)](image)

Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to Down-counting mode.

In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit
(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to Center-aligned mode (up/down-counting).

Figure 517 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Hints on using center-aligned mode:
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit.
in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.

- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

**Dithering mode**

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 518* presents the dithering principle applied to four consecutive PWM cycles.

*Figure 518. Dithering principle*

When the dithering mode is enabled, the register coding is changed as following (see *Figure 519* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.
Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 519. Data format and register coding in dithering mode

The minimum frequency is given by the following formula:

\[
\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \\Rightarrow F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\text{Max Resolution}}
\]

Dithering mode disabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode (16-bit timer): \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \)

Dithering mode (32-bit timer): \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{268435454 + \frac{15}{16}} \)

Note: For 16-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFFE in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).
For 32-bit timers, the maximum TIMx_ARR and TIMxCCRy values are limited to
0xFFFFFEEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on Figure 520 and Figure 521, the dithering mode is used to increase the PWM resolution.

Figure 520. PWM resolution vs frequency (16-bit mode)

Figure 521. PWM resolution vs frequency (32-bit mode)

The duty cycle and/or period changes are spread over 16 consecutive periods, as described in Figure 522.
The autoreload and compare values increments are spread following specific patterns described in Table 433. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.
The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in Figure 523.

### Table 433. CCR and ARR register change dithering pattern

<table>
<thead>
<tr>
<th>LSB value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1110</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1111</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>

**Figure 523. Dithering effect on duty cycle in center-aligned PWM mode**
Table 434 shows how the dithering pattern is added in center-aligned PWM mode.

Table 434. CCR register change dithering pattern in center-aligned PWM mode

<table>
<thead>
<tr>
<th>LSB value</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td></td>
<td>Up</td>
</tr>
<tr>
<td>0000</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
</tr>
<tr>
<td>1110</td>
<td>+1</td>
</tr>
<tr>
<td>1111</td>
<td>+1</td>
</tr>
</tbody>
</table>

42.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.
Figure 524 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

**Figure 524. Generation of two phase-shifted PWM signals with 50% duty cycle**

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_oc1refc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR1=0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR2=8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tim_oc3refc</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR3=3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CCR4=5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**42.4.13 Combined PWM mode**

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 525 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.
42.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to Section 42.3: TIM2/TIM3/TIM4/TIM5 implementation.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etrf_in after the filter) by configuring the OCCE bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in Figure 526.
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etrf must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIM1_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application’s needs.

Figure 527 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.
Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

42.4.15 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 528. Example of One-pulse mode

For example if the user wants to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
- The tDELAY is defined by the value written in the TIMx_CCR1 register.
- The tPULSE is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2, CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

**Particular case: tim_ocx fast enable:**

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay tDELAY min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 42.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in Section 42.4.15:
- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

**Note:** In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

### 42.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on Figure 530.
**General-purpose timers (TIM2/TIM3/TIM4/TIM5)**

**Figure 530. Pulse generator circuitry**

![Pulse generator circuitry diagram](image)

**Figure 531. Pulse generation on compare event, for edge-aligned and encoder modes**

![Pulse generation diagram](image)

This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.
The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

\[ t_{PW} = PW[7:0] \times t_{PWG} \]

where \( t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim\_ker\_ck} \)

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

**Note:** If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 532).

![Figure 532. Extended pulse width in case of concurrent triggers](MSv62348V1)

### 42.4.18 Encoder interface mode

**Quadrature encoder**

To select Encoder interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_t1 edges only, SMS = 0010 if it is counting on tim_t2 edges only and SMS = 0011 if it is counting on both tim_t1 and tim_t2 edges.

Select the tim_t1 and tim_t2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_t1 and tim_t2 are used to interface to an incremental encoder. Refer to Table 435. The counter is clocked by each valid transition on tim_t1fp1 or tim_t2fp2 (tim_t1 and tim_t2 after input filter and polarity selection, tim_t1fp1 = tim_t1 if not filtered and not inverted, tim_t2fp2 = tim_t2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_t1 or
tim_t12), whatever the counter is counting on tim_t1 only, tim_t12 only or both tim_t1 and tim_t12.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder’s position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_t1 and tim_t12 do not switch at the same time.

Table 435. Counting direction versus encoder signals(CC1P = CC2P = 0)

<table>
<thead>
<tr>
<th>Active edge</th>
<th>SMS[3:0]</th>
<th>Level on opposite signal (tim_t1fp1 for tim_t1, tim_t12fp2 for tim_t1)</th>
<th>tim_t1fp1 signal</th>
<th>tim_t12fp2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Counting on tim_t1 only x1 mode</td>
<td>1110</td>
<td>High, Down, Up</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, No count, No count</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>Counting on tim_t1 only x2 mode</td>
<td>0001</td>
<td>High, Down, Up</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, Up, Down</td>
<td>No count</td>
<td>Down</td>
</tr>
<tr>
<td>Counting on tim_t12 only x2 mode</td>
<td>0010</td>
<td>High, No count, No count</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, No count, No count</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td>Counting on tim_t1 and tim_t12 x4 mode</td>
<td>0011</td>
<td>High, Down, Up</td>
<td>Up</td>
<td>Down</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Low, Up, Down</td>
<td>Down</td>
<td>Up</td>
</tr>
</tbody>
</table>

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder’s differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

**Figure 533** gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are
selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

**Figure 533. Example of counter operation in encoder interface mode**

![Counter Operation Diagram](MSv62349V1)

*Figure 534* gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

**Figure 534. Example of encoder interface mode with tim_ti1fp1 polarity inverted**

![Polarity Inverted Diagram](MSv62350V1)
Figure 535 shows the timer counter value during a speed reversal, for various counting modes.

**Figure 535. Quadrature encoder counting modes**

The timer, when configured in Encoder Interface mode provides information on the sensor’s current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register’s bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

**Clock plus direction encoder mode**

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

In the “clock plus direction” mode shown on Figure 536, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.
This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- **1010**: x2 mode, the counter is updated on both rising and falling edges of the clock.
- **1011**: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

**Figure 536. Direction plus clock encoder mode**

**Directional clock encoder mode**

In the “directional clock” mode on Figure 537, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- **1100**: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to Figure 537) and CCxP = 1 corresponds to low-level idle state (refer to Figure 538).
- **1101**: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to Figure 537), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to Figure 538).
Figure 537. Directional clock encoder mode (CC1P = CC2P = 0)

Figure 538. Directional clock encoder mode (CC1P = CC2P = 1)
Table 436 details how the directional clock mode operates, for any input transition.

### Table 436. Counting direction versus encoder signals and polarity settings

<table>
<thead>
<tr>
<th>Directional clock mode</th>
<th>SMS[3:0]</th>
<th>Level on opposite signal (tim_t1fp1 for tim_t1, tim_t12fp2 for tim_t1)</th>
<th>tim_t1fp1 signal</th>
<th>tim_t12fp2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>x2 mode</td>
<td>1100</td>
<td>High</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>CCxP = 0</td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x2 mode</td>
<td>1100</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>CCxP = 1</td>
<td></td>
<td>Low</td>
<td>Down</td>
<td>Down</td>
</tr>
<tr>
<td>x1 mode</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>CCxP = 0</td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>x1 mode</td>
<td>1101</td>
<td>High</td>
<td>No count</td>
<td>No count</td>
</tr>
<tr>
<td>CCxP = 1</td>
<td></td>
<td>Low</td>
<td>No count</td>
<td>No count</td>
</tr>
</tbody>
</table>

**Index input**

The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per Figure 539:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.
The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on Figure 540.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. Figure 541 shows at which position is the index generated, for a simplistic example (an encoder providing four edges per mechanical rotation).
Figure 541. Index generation for IPOS[1:0] = 11

Figure 542 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.
The arrows are indicating on which transition is the index event interrupt generated.

Figure 542. Counter reading with index gated on channel A (IPOS[1:0] = 11)

Figure 543 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.
Figure 543. Counter reading with index ungated (IPOS[1:0] = 00)

Figure 544 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 544. Counter reading with index gated on channel A and B

Figure 545 and Figure 546 detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.
Figure 545. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)
Figure 546. Counter reset Narrow index pulse (closer view, ARR = 0x07)

<table>
<thead>
<tr>
<th>Channel A</th>
<th>Channel B</th>
<th>Index</th>
<th>DIR bit</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5 6 7 0 1 2 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4 5 6 0 1 2 3</td>
</tr>
</tbody>
</table>
**Figure 547** shows how the index is managed in x1 and x2 modes.

![Index behavior in x1 and x2 mode (IPOS[1:0] = 01)](image)

**Directional index sensitivity**

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

**Figure 548** shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

![Directional index sensitivity](image)
Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on Figure 549. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

**Figure 549. Counter reset as function of FIDX bit setting**

Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on Figure 550.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:
- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

**Figure 550. Index blanking**
Index management in nonquadrature mode

*Figure 551* and *Figure 552* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:
- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

**Figure 551. Index behavior in clock + direction mode, IPOS[0] = 1**

**Figure 552. Index behavior in directional clock mode, IPOS[0] = 1**

Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 553*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR
status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

**Figure 553. State diagram for quadrature encoded signals**

For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on Figure 554.

**Figure 554. Up-counting encoder error detection**

<table>
<thead>
<tr>
<th>Channel A</th>
<th>Channel B</th>
<th>Index</th>
<th>IERRF</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Error detected | Abort (index detection)

<table>
<thead>
<tr>
<th>Channel A</th>
<th>Channel B</th>
<th>Index</th>
<th>IERRF</th>
<th>Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

Error detected | Error asserted
In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on Figure 555, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

**Figure 555. Down-counting encode error detection**

An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

**Functional encoder interrupts**

The following interrupts are also available in encoder mode:

- **Direction change**: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.

- **Index event**: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on Figure 556.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMPS bit in the TIMx_SMCR register.

- SMPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMPS = 1: the transfer is triggered by the index event.

Figure 556. Encoder mode change with preload transferred on update (SMPS = 0)

Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

42.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.
This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

42.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register’s bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter’s most significant bit is only accessible in write mode).

42.4.21 Timer input XOR function

The TI1S bit in the TIM1xx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in Section 41.3.29: Interfacing with Hall sensors.

42.4.22 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the
trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

![Control circuit in reset mode](MSv62361V1)

**Figure 557. Control circuit in reset mode**

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on TIM\_ti2 input:

1. Configure the channel 2 to detect rising edges on TIM\_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIMx\_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx\_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS = 110 in TIMx\_SMCR register. Select TIM\_ti2 as the input source by writing TS = 00110 in TIMx\_SMCR register.

When a rising edge occurs on TIM\_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on TIM\_ti2 and the actual start of the counter is due to the resynchronization circuit on TIM\_ti2 input.
Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.

2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01 in TIMx_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).

3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.
42.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in Master mode, it can reset, start, stop, or clock the counter of another timer configured in Slave mode.

*Figure 561* and *Figure 562* show examples of master/slave timer connections.
Figure 562. Master/slave connection example with 1 channel only timers

Note: The timers with one channel only (see Figure 562) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals). The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger. For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to Figure 561. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to Figure 561 for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck (f_{tim_cnt_ck} = f_{tim_ker_ck}/3).
1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).

2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).

3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).

4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).

5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).

6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

**Note:** The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

**Figure 563. Gating TIM_slv with tim_oc1ref of TIM_mstr**

In the example in *Figure 563*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.
In the next example (refer to Figure 564), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0x7E. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0x7E by writing 0x7E in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 564. Gating TIM_slv with Enable of TIM_mstr

Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to Figure 561 for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck (f_{tim_ker_ck} = f_{tim_ker_ck}/3).
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

**Figure 565. Triggering TIM_slv with update of TIM_mstr**

As in the previous example, both counters can be initialized before starting counting. **Figure 566** shows the behavior with the same configuration as in **Figure 565** but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

**Figure 566. Triggering TIM_slv with Enable of TIM_mstr**
Starting two timers synchronously in response to an external trigger

In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to Figure 561 for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 567. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.
Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

42.4.25 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address, i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see Section 42.5.23: TIMx DMA control register (TIMx_DCR)(x = 2 to 5) for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.

2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.

3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).

4. Enable TIMx.

5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let’s take the example of a buffer in the RAM containing data1, data2, data3, data4, data5,
and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

42.4.26 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in Table 437.

<table>
<thead>
<tr>
<th>DMA request signal</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Capture/compare 1</td>
<td>CC1DE</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Capture/compare 2</td>
<td>CC2DE</td>
</tr>
<tr>
<td>tim_cc3_dma</td>
<td>Capture/compare 3</td>
<td>CC3DE</td>
</tr>
<tr>
<td>tim_cc4_dma</td>
<td>Capture/compare 4</td>
<td>CC4DE</td>
</tr>
<tr>
<td>tim_trgi_dma</td>
<td>Trigger</td>
<td>TDE</td>
</tr>
</tbody>
</table>

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

42.4.27 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

42.4.28 TIM2/TIM3/TIM4/TIM5 low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>
42.4.29 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in Table 439.

Table 439. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_UP</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_CC</td>
<td>Capture/compare 1</td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 2</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 3</td>
<td>CC3IF</td>
<td>CC3IE</td>
<td>write 0 in CC3IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 4</td>
<td>CC4IF</td>
<td>CC4IE</td>
<td>write 0 in CC4IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TRG</td>
<td>Trigger</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_DIR _IDX</td>
<td>Index</td>
<td>IDXF</td>
<td>IDXIE</td>
<td>write 0 in IDXF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Direction</td>
<td>DIRF</td>
<td>DIRIE</td>
<td>write 0 in DIRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_IERR</td>
<td>Index Error</td>
<td>IERRF</td>
<td>IERRIE</td>
<td>write 0 in IERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM_TER</td>
<td>Transition Error</td>
<td>TERRF</td>
<td>TERRIE</td>
<td>write 0 in TERRF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
42.5 **TIM2/TIM3/TIM4/TIM5 registers**

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

42.5.1 **TIMx control register 1 (TIMx_CR1)(x = 2 to 5)**

Address offset: 0x000

Reset value: 0x0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled
1: Dithering enabled

*Note: The DITHEN bit can only be modified when CEN bit is reset.*

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: tDTS = tim_ker_ck
01: tDTS = 2 × tim_ker_ck
10: tDTS = 4 × tim_ker_ck
11: Reserved

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

*Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)*
Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
*Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.*

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
*Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*
CEN is cleared automatically in one-pulse mode, when an update event occurs.

### 42.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ST**

RM0477 Rev 6 1849/3791
Bits 31:26  Reserved, must be kept at reset value.

Bits 24:8  Reserved, must be kept at reset value.

Bit 7  **TI1S**: tim_t1 selection

0: The \( \text{tim_t1}_\text{in}[15:0] \) multiplexer output is to \( \text{tim_t1}_\text{input} \) input
1: The \( \text{tim_t1}_\text{in}[15:0], \text{tim_t2}_\text{in}[15:0] \) and \( \text{tim_t3}_\text{in}[15:0] \) multiplexers outputs are XORed
and connected to the \( \text{tim_t1}_\text{input} \). See also **Section 41.3.29: Interfacing with Hall sensors**.

Bits 25, 6, 5, 4  **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for
synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If

the reset is generated by the trigger input (slave mode controller configured in reset
mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is

useful to start several timers at the same time or to control a window in which a slave
timer is enabled. The Counter Enable signal is generated by a logic AND between
CEN control bit and the trigger input when configured in gated mode.

When the Counter Enable signal is controlled by the trigger input, there is a delay on
tim_trgo, except if the master/slave mode is selected (see the MSM bit description in
TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a

master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to

be set (even if it was already high), as soon as a capture or a compare match
occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)
0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)
0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)
0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo).

This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011,

1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to
unexpected behavior.

Others: Reserved

**Note**: The clock of the slave timer or ADC must be enabled prior to receive events from the
master timer, and must not be changed on-the-fly while triggers are received from the
master timer.

Bit 3  **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bits 2:0  Reserved, must be kept at reset value.
42.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x008
Reset value: 0x0000 0000

| Bit 31:26 | Reserved, must be kept at reset value. |
| Bit 25   | **SMSPS**: SMS preload source |
|          | This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active |
|          | 0: The transfer is triggered by the Timer’s Update event |
|          | 1: The transfer is triggered by the Index event |
| Bit 24   | **SMSPE**: SMS preload enable |
|          | This bit selects whether the SMS[3:0] bitfield is preloaded |
|          | 0: SMS[3:0] bitfield is not preloaded |
|          | 1: SMS[3:0] preload is enabled |
| Bit 15   | **ETP**: External trigger polarity |
|          | This bit selects whether tim_etr_in or tim_etr_in is used for trigger operations |
|          | 0: tim_etr_in is non-inverted, active at high level or rising edge |
|          | 1: tim_etr_in is inverted, active at low level or falling edge |
| Bit 14   | **ECE**: External clock enable |
|          | This bit enables External clock mode 2. |
|          | 0: External clock mode 2 disabled |
|          | 1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal. |
|          | **Note**: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111). |
|          | It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111). |
|          | If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf. |
| Bit 13:12| **ETPS[1:0]**: External trigger prescaler |
|          | External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in. |
|          | 00: Prescaler OFF |
|          | 01: tim_etrp frequency divided by 2 |
|          | 10: tim_etrp frequency divided by 4 |
|          | 11: tim_etrp frequency divided by 8 |
Bits 11:8 **ETF[3:0]: External trigger filter**

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- **0000**: No filter, sampling is done at fDTS
- **0001**: fSAMPLING = ftim_ker_ck, N = 2
- **0010**: fSAMPLING = ftim_ker_ck, N = 4
- **0011**: fSAMPLING = ftim_ker_ck, N = 8
- **0100**: fSAMPLING = fDTS/2, N = 6
- **0101**: fSAMPLING = fDTS/2, N = 8
- **0110**: fSAMPLING = fDTS/4, N = 6
- **0111**: fSAMPLING = fDTS/4, N = 8
- **1000**: fSAMPLING = fDTS/8, N = 6
- **1001**: fSAMPLING = fDTS/8, N = 8
- **1010**: fSAMPLING = fDTS/16, N = 5
- **1011**: fSAMPLING = fDTS/16, N = 6
- **1100**: fSAMPLING = fDTS/16, N = 8
- **1101**: fSAMPLING = fDTS/32, N = 5
- **1110**: fSAMPLING = fDTS/32, N = 6
- **1111**: fSAMPLING = fDTS/32, N = 8

Bit 7 **MSM: Master/Slave mode**

- **0**: No action
- **1**: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]:** Trigger selection
   This bitfield selects the trigger input to be used to synchronize the counter.
   00000: Internal trigger 0 (tim_itr0)
   00001: Internal trigger 1 (tim_itr1)
   00010: Internal trigger 2 (tim_itr2)
   00011: Internal trigger 3 (tim_itr3)
   00100: tim_ti1 edge detector (tim_ti1f_ed)
   00101: Filtered timer input 1 (tim_ti1fp1)
   00110: Filtered timer input 2 (tim_ti2fp2)
   00111: External trigger input (tim_etrf)
   01000: Internal trigger 4 (tim_itr4)
   01001: Internal trigger 5 (tim_itr5)
   01010: Internal trigger 6 (tim_itr6)
   01011: Internal trigger 7 (tim_itr7)
   01100: Internal trigger 8 (tim_itr8)
   01101: Internal trigger 9 (tim_itr9)
   01110: Internal trigger 10 (tim_itr10)
   01111: Internal trigger 11 (tim_itr11)
   10000: Internal trigger 12 (tim_itr12)
   10001: Internal trigger 13 (tim_itr13)
   10010: Internal trigger 14 (tim_itr14)
   10011: Internal trigger 15 (tim_itr15)
   Others: Reserved
   See Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation details.

   **Note:** These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3  **OCCS:** OCREF clear selection
   This bit is used to select the OCREF clear source
   0: tim_ocref_clr_int is connected to the tim_ocref_clr input
   1: tim_ocref_clr_int is connected to tim_etrf

   **Note:** If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 42.3: TIM2/TIM3/TIM4/TIM5 implementation.
Bits 16, 2, 1, 0  **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etra_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).
0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.
0001: Encoder mode 1 - Counter counts up/down on tim_t1f1fp1 edge depending on tim_t1f2fp2 level.
0010: Encoder mode 2 - Counter counts up/down on tim_t1f2fp2 edge depending on tim_t1f1fp1 level.
0011: Encoder mode 3 - Counter counts up/down on both tim_t1f1fp1 and tim_t1f2fp2 edges depending on the level of the other input.
0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.
0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.
0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.
1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
1010: Encoder mode: Clock plus direction, x2 mode.
1011: Encoder mode: Clock plus direction, x1 mode, tim_t1f2fp2 edge sensitivity is set by CC2P.
1100: Encoder mode: Directional clock, x2 mode.
1101: Encoder mode: Directional clock, x1 mode, tim_t1f1fp1 and tim_t1f2fp2 edge sensitivity is set by CC1P and CC2P.
1110: Quadrature encoder mode: x1 mode, counting on tim_t1f1fp1 edges only, edge sensitivity is set by CC1P.
1111: Quadrature encoder mode: x1 mode, counting on tim_t1f2fp2 edges only, edge sensitivity is set by CC2P.

**Note:** The gated mode must not be used if tim_t1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_t1f_ed outputs 1 pulse for each transition on tim_t1f, whereas the gated mode checks the level of the trigger signal.

**Note:** The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.
### 42.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 31    | TERRIE: Transition error interrupt enable  
            0: Transition error interrupt disabled  
            1: Transition error interrupt enabled |
| Bit 30    | IERRIE: Index error interrupt enable  
          0: Index error interrupt disabled  
          1: Index error interrupt enabled |
| Bit 29    | DIRIE: Direction change interrupt enable  
          0: Direction change interrupt disabled  
          1: Direction change interrupt enabled |
| Bit 28    | IDXIE: Index interrupt enable  
         0: Index interrupt disabled  
         1: Index interrupt enabled |
| Bits 27:20 | Reserved, must be kept at reset value. |
| Bit 27    | TDE: Trigger DMA request enable  
           0: Trigger DMA request disabled  
           1: Trigger DMA request enabled |
| Bit 26    | CC4DE: Capture/Compare 4 DMA request enable  
            0: CC4 DMA request disabled  
            1: CC4 DMA request enabled |
| Bit 25    | CC3DE: Capture/Compare 3 DMA request enable  
           0: CC3 DMA request disabled  
           1: CC3 DMA request enabled |
| Bit 24    | CC2DE: Capture/Compare 2 DMA request enable  
           0: CC2 DMA request disabled  
           1: CC2 DMA request enabled |
| Bit 23    | CC1DE: Capture/Compare 1 DMA request enable  
           0: CC1 DMA request disabled  
           1: CC1 DMA request enabled |
| Bit 22    | UDE: Update DMA request enable  
         0: Update DMA request disabled  
         1: Update DMA request enabled |

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

---

**Notes:**
- The TIMx DMA/Interrupt enable register (TIMx_DIER) is used to enable or disable various DMA and interrupt requests for the TIMx timer. The register is accessed through a specific address offset and has a reset value to ensure proper functionality.
- Each bit in the register controls a specific DMA or interrupt request, such as transition error, index error, direction change, and others.
- The register is configured in read-write mode, allowing users to read the current state and write new configurations as needed.
- Proper configuration of these bits is crucial for ensuring the correct operation of the timer and its associated peripherals in applications requiring precise timing and event handling.
General-purpose timers (TIM2/TIM3/TIM4/TIM5)

42.5.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:24</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>
| Bit 23 | **TERRF**: Transition error interrupt flag  
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.  
0: No encoder transition error has been detected.  
1: An encoder transition error has been detected |
| Bit 22 | **IERRF**: Index error interrupt flag  
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.  
0: No index error has been detected.  
1: An index error has been detected |
Bit 21  **DIRF**: Direction change interrupt flag
   This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
   0: No direction change
   1: Direction change

Bit 20  **IDXF**: Index interrupt flag
   This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
   0: No index event occurred.
   1: An index event has occurred

Bits 19:13  Reserved, must be kept at reset value.

Bit 12  **CC4OF**: Capture/Compare 4 overcapture flag
   refer to CC1OF description

Bit 11  **CC3OF**: Capture/Compare 3 overcapture flag
   refer to CC1OF description

Bit 10  **CC2OF**: Capture/compare 2 overcapture flag
   refer to CC1OF description

Bit 9  **CC1OF**: Capture/Compare 1 overcapture flag
   This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
   0: No overcapture has been detected.
   1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7  Reserved, must be kept at reset value.

Bit 6  **TIF**: Trigger interrupt flag
   This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
   0: No trigger event occurred.
   1: Trigger interrupt pending.

Bit 5  Reserved, must be kept at reset value.

Bit 4  **CC4IF**: Capture/Compare 4 interrupt flag
   Refer to CC1IF description

Bit 3  **CC3IF**: Capture/Compare 3 interrupt flag
   Refer to CC1IF description
Bit 2  **CC2IF**: Capture/Compare 2 interrupt flag  
Refer to CC1IF description

Bit 1  **CC1IF**: Capture/compare 1 interrupt flag  
This flag is set by hardware. It is cleared by software (input capture or output compare mode)  
or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred

**If channel CC1 is configured as output:** this flag is set when the content of the counter  
TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of  
TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the  
counter overflow (in up-counting and up/down-counting modes) or underflow (in down- 
counting mode). There are three possible options for flag setting in center-aligned mode,  
refer to the CMS bits in the TIMx_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured  
in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity  
defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0  **UIF**: Update interrupt flag  
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:  
At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and  
UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description),  
if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

### 42.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014
Reset value: 0x0000

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 15| 14| 13| 12| 11| 10| 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|   |   |   |   |   |   | TG| CC4G| CC3G| CC2G| CC1G| UG |
| w | w | w | w | w | w |

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

Bit 6  **TG**: Trigger generation  
This bit is set by software in order to generate an event, it is automatically cleared by  
hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if  
enabled.

Bit 5  Reserved, must be kept at reset value.

Bit 4  **CC4G**: Capture/compare 4 generation  
Refer to CC1G description

Bit 3  **CC3G**: Capture/compare 3 generation  
Refer to CC1G description
42.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Input capture mode

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:12: **IC2F[3:0]**: Input capture 2 filter
- Bits 11:10: **IC2PSC[1:0]**: Input capture 2 prescaler
Bits 9:8 **CC2S[1:0]:** Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
- 00: CC2 channel is configured as output.
- 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.
- 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.
- 11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*

Bits 7:4 **IC1F[3:0]:** Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
- 0000: No filter, sampling is done at f_DTS
- 0001: f_SAMPLING = f_tim_ker_ck, N = 2
- 0010: f_SAMPLING = f_tim_ker_ck, N = 4
- 0011: f_SAMPLING = f_tim_ker_ck, N = 8
- 0100: f_SAMPLING = f_DTS/2, N = 6
- 0101: f_SAMPLING = f_DTS/2, N = 8
- 0110: f_SAMPLING = f_DTS/4, N = 6
- 0111: f_SAMPLING = f_DTS/4, N = 8
- 1000: f_SAMPLING = f_DTS/8, N = 6
- 1001: f_SAMPLING = f_DTS/8, N = 8
- 1010: f_SAMPLING = f_DTS/16, N = 5
- 1011: f_SAMPLING = f_DTS/16, N = 6
- 1100: f_SAMPLING = f_DTS/16, N = 8
- 1101: f_SAMPLING = f_DTS/32, N = 5
- 1110: f_SAMPLING = f_DTS/32, N = 6
- 1111: f_SAMPLING = f_DTS/32, N = 8

Bits 3:2 **IC1PSC[1:0]:** Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]:** Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*
42.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

### Output compare mode

| Bit 31-25 | Reserved, must be kept at reset value. |
| Bit 23-17 | Reserved, must be kept at reset value. |
| Bit 15 | OC2CE: Output compare 2 clear enable |
| Bit 24, 14-12 | OC2M[3:0]: Output compare 2 mode refer to OC1M description on bits 6:4 |
| Bit 11 | OC2PE: Output compare 2 preload enable |
| Bit 10 | OC2FE: Output compare 2 fast enable |
| Bit 9-8 | CC2S[1:0]: Capture/Compare 2 selection This bitfield defines the direction of the channel (input/output) as well as the used input. 00: CC2 channel is configured as output 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1 11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register) |
| Bit 7 | OC1CE: Output compare 1 clear enable 0: tim_oc1ref is not affected by the tim_ocref_clr_int input 1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input |

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>OC2M[3]</td>
</tr>
<tr>
<td>25</td>
<td>Res.</td>
</tr>
<tr>
<td>24</td>
<td>Res.</td>
</tr>
<tr>
<td>23</td>
<td>Res.</td>
</tr>
<tr>
<td>22</td>
<td>Res.</td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
</tr>
<tr>
<td>19</td>
<td>Res.</td>
</tr>
<tr>
<td>18</td>
<td>Res.</td>
</tr>
<tr>
<td>17</td>
<td>OC1M[3]</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>OC2CE</td>
</tr>
<tr>
<td>14</td>
<td>OC2M[2:0]</td>
</tr>
<tr>
<td>13</td>
<td>OC2PE</td>
</tr>
<tr>
<td>12</td>
<td>OC2FE</td>
</tr>
<tr>
<td>11</td>
<td>CC2S[1:0]</td>
</tr>
<tr>
<td>10</td>
<td>OC1CE</td>
</tr>
<tr>
<td>9</td>
<td>OC1M[2:0]</td>
</tr>
<tr>
<td>8</td>
<td>OC1PE</td>
</tr>
<tr>
<td>7</td>
<td>OC1FE</td>
</tr>
<tr>
<td>6</td>
<td>CC1S[1:0]</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCMR).
Bits 16, 6:4 **OC1M[3:0]:** Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT> TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, channel 1 is active as long as TIMx_CNT> TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

**Note:** In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.
42.5.9 **TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)**

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

**Input capture mode**

- Bits 31:16: Reserved, must be kept at reset value.
- Bits 15:12: *IC4F[3:0]*: Input capture 4 filter
- Bits 11:10: *IC4PSC[1:0]*: Input capture 4 prescaler
Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** **CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).**

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

**Note:** **CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).**

**42.5.10 TIMx capture/compare mode register 2 [alternate]**
(TMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Output compare mode**

Bits 31:25 Reserved, must be kept at reset value.

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

Bit 15 **OC4CE:** Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]:** Output compare 4 mode
Refer to OC3M[3:0]

Bit 11 **OC4PE:** Output compare 4 preload enable

Bit 10 **OC4FE:** Output compare 4 fast enable
Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
   
   This bitfield defines the direction of the channel (input/output) as well as the used input.
   
   00: CC4 channel is configured as output
   01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
   10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
   11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

   **Note:** CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable
Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs. (this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

**Note:** In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from "frozen" mode to "PWM" mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.
Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

*Note*: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

### 42.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CC4NP</strong></td>
<td><strong>CC4P</strong></td>
<td><strong>CC4E</strong></td>
<td><strong>CC3NP</strong></td>
<td><strong>CC3P</strong></td>
<td><strong>CC3E</strong></td>
<td><strong>CC2NP</strong></td>
<td><strong>CC2P</strong></td>
<td><strong>CC2E</strong></td>
<td><strong>CC1NP</strong></td>
<td><strong>CC1P</strong></td>
<td><strong>CC1E</strong></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 15 **CC4NP**: Capture/Compare 4 output Polarity.
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: Capture/Compare 4 output Polarity.
Refer to CC1P description

Bit 12 **CC4E**: Capture/Compare 4 output enable.
Refer to CC1E description

Bit 11 **CC3NP**: Capture/Compare 3 output Polarity.
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: Capture/Compare 3 output Polarity.
Refer to CC1P description

Bit 8 **CC3E**: Capture/Compare 3 output enable.
Refer to CC1E description

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity.
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity.
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable.
Refer to CC1E description
Bit 3 **CC1NP**: Capture/Compare 1 output Polarity.

**CC1 channel configured as output**: CC1NP must be kept cleared in this case.

**CC1 channel configured as input**: This bit is used in conjunction with CC1P to define Tim_T1fp1/Tim_T2fp1 polarity. Refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

- 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
- 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of T11FP1 and T12FP1 for trigger or capture operations.

- **CC1NP = 0, CC1P = 0**: non-inverted/rising edge. The circuit is sensitive to TixFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TixFP1 is not inverted (trigger operation in gated mode or encoder mode).
- **CC1NP = 0, CC1P = 1**: inverted/falling edge. The circuit is sensitive to TixFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TixFP1 is inverted (trigger operation in gated mode or encoder mode).
- **CC1NP = 1, CC1P = 1**: non-inverted/both edges. The circuit is sensitive to both TixFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TixFP1 is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.
- **CC1NP = 1, CC1P = 0**: this configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

- 0: Capture mode disabled / OC1 is not active
- 1: Capture mode enabled / OC1 signal is output on the corresponding output pin

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td>1</td>
<td>Output enabled (tim_ocx = tim_ocxref + Polarity)</td>
</tr>
</tbody>
</table>

**Note:** The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.
42.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5)
Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY_CNT[31]: Value depends on UIFREMAP in TIMx_CR1.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>If UIFREMAP = 0</td>
</tr>
<tr>
<td></td>
<td>CNT[31]: Most significant bit of counter value</td>
</tr>
<tr>
<td></td>
<td>If UIFREMAP = 1</td>
</tr>
<tr>
<td></td>
<td>UIFCPY: UIF Copy</td>
</tr>
<tr>
<td></td>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register</td>
</tr>
</tbody>
</table>

Bits 30:0 CNT[30:0]: Least significant part of counter value

Non-dithering mode (DITHEN = 0)
The register holds the counter value.

Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

42.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5)
Address offset: 0x028
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The counter clock frequency tim_cnt_ck is equal to ftim_psc_ck / (PSC[15:0] + 1).</td>
</tr>
<tr>
<td></td>
<td>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).</td>
</tr>
</tbody>
</table>
42.5.14  TIMx autoreload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x02C  
Reset value: 0xFFFF FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  ARR[31:0]: Autoreload value
ARR is the value to be loaded in the actual autoreload register.
Refer to the Section 42.4.3: Time-base unit for more details about ARR update and behavior.
The counter is blocked while the autoreload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the autoreload value.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

42.5.15  TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x034  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value  
*If channel CC1 is configured as output:*  
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit O1C1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.  
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.  
**Non-dithering mode (DITHEN = 0)**  
The register holds the compare value.  
**Dithering mode (DITHEN = 1)**  
The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.  
*If channel CC1 is configured as input:*  
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.  
**Non-dithering mode (DITHEN = 0)**  
The register holds the capture value.  
**Dithering mode (DITHEN = 1)**  
The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

### 42.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x038  
Reset value: 0x0000 0000

| CCR2[31:16] | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| `rw`        | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` |
| CCR2[15:0]  | 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
| `rw`        | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` | `rw` |
Bits 31:0  **CCR2[31:0]**: Capture/compare 2 value

**If channel CC2 is configured as output:**
CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

**If channel CC2 is configured as input:**
CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

### 42.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x03C
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**CCR3[31:16]**

**CCR3[15:0]**
Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

**If channel CC3 is configured as output:**
CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

**If channel CC3 is configured as input:**
CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

### 42.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

**Address offset:** 0x040

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CCR4[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CCR4[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:0  **CCR4[31:0]**: Capture/compare 4 value

**If channel CC4 is configured as output:**
CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

**Non-dithering mode (DITHEN = 0)**
The register holds the compare value.

**Dithering mode (DITHEN = 1)**
The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

**If channel CC4 is configured as input:**
CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**
The register holds the capture value.

**Dithering mode (DITHEN = 1)**
The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

### 42.5.19  **TIMx timer encoder control register (TIMx_ECR)**(x = 2 to 5)

Address offset: 0x058
Reset value: 0x0000 0000

| Bits 31:27 | Reserved, must be kept at reset value. |
| Bits 26:24 | **PWPRSC[2:0]**: Pulse width prescaler |
| **This bitfield sets the clock prescaler for the pulse generator, as following:** |
| | \( t_{PWG} = \left(2^{(PWPRSC[2:0])}\right) \times t_{tim\_ker\_ck} \) |
| Bits 23:16 | **PW[7:0]**: Pulse width |
| **This bitfield defines the pulse duration, as following:** |
| | \( t_{PW} = PW[7:0] \times t_{PWG} \) |
| Bits 15:8 | Reserved, must be kept at reset value. |
### 42.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

**Address offset:** 0x05C  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.
General-purpose timers (TIM2/TIM3/TIM4/TIM5) RM0477

42.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

ETRSEL[3:2]:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ETRSEL[1:0]:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
... 1111: tim_ti4_in15
Refer to Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

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

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
... 1111: tim_ti3_in15
Refer to Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

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

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
... 1111: tim_ti2_in15
Refer to Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
... 1111: tim_ti1_in15
Refer to Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.
Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]: etr_in source selection**
These bits select the etr_in input source.
- 0000: tim_et0: TIMx_ETR input
- 0001: tim_et1
- ...
- 1111: tim_et15
Refer to Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

### 42.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>Reserved</td>
<td>0000000000000000</td>
</tr>
<tr>
<td>19-16</td>
<td>OCRSEL[2:0]: ocref_clr source selection</td>
<td>0000000000000000</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]: ocref_clr source selection**
These bits select the ocref_clr input source.
- 000: tim_ocref_clr0
- 001: tim_ocref_clr1
- ...
- 111: tim_ocref_clr7
Refer to Section 42.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.
42.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
- 0000: Reserved
- 0001: Update
- 0010: CC1
- 0011: CC2
- 0100: CC3
- 0101: CC4
- 0110: COM
- 0111: Trigger
- Others: reserved

Bits 15:13 Reserved, must be kept at reset value.
42.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0
Reset value: 0x0000 0000
Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

\[(\text{TIMx} \_\text{CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4\]

where **TIMx\_CR1 address** is the address of the control register 1, **DBA** is the DMA base address configured in **TIMx\_DCR** register, **DMA index** is automatically controlled by the DMA transfer, and ranges from 0 to **DBL** (DBL configured in **TIMx\_DCR**).
### 42.5.25 TIMx register map

TIMx registers are mapped as described in the table below.

Table 441. TIM2/TIM3/TIM4/TIM5 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>TIMx_CR1</th>
<th>TIMx_CR2</th>
<th>TIMx_SMCR</th>
<th>TIMx_DIER</th>
<th>TIMx_SR</th>
<th>TIMx_EGR</th>
<th>TIMx_CCMR1</th>
<th>TIMx_CCMR2</th>
<th>TIMx_CCER</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>TIMx_CR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x004</td>
<td>TIMx_CR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x004</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x008</td>
<td>TIMx_SMCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x008</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x00C</td>
<td>TIMx_DIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x00C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x010</td>
<td>TIMx_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x010</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x014</td>
<td>TIMx_EGR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x014</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x018</td>
<td>TIMx_CCMR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x018</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x01C</td>
<td>TIMx_CCMR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x01C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x020</td>
<td>TIMx_CCER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value</td>
<td>0x020</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
### General-purpose timers (TIM2/TIM3/TIM4/TIM5) RM0477

#### Table 441. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x024</td>
<td>Timx_CNT</td>
<td>0x028</td>
<td>Timx_PSC</td>
<td>0x02C</td>
<td>Timx_ARR</td>
<td>0x030</td>
<td>Timx_CCR1</td>
<td>0x034</td>
<td>Timx_CCR1</td>
</tr>
<tr>
<td>Reset value 0x00000000 0x00000000</td>
<td>Reset value 0x00000000</td>
<td>Reset value 11111111</td>
<td>Reset value 0000000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td></td>
<td>0x038</td>
<td>Timx_CCR2</td>
<td>0x03C</td>
<td>Timx_CCR3</td>
<td>0x040</td>
<td>Timx_CCR4</td>
<td>0x044</td>
<td>Reserved</td>
</tr>
<tr>
<td>Reset value 0x00000000 0x00000000</td>
<td>Reset value 0000000000000000</td>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x03C</td>
<td></td>
<td>0x058</td>
<td>Timx_ECR</td>
<td>0x05C</td>
<td>Timx_TISEL</td>
<td>0x060</td>
<td>Timx_AF1</td>
<td>0x064</td>
<td>Timx_AF2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PW[7:0]</td>
<td></td>
<td>TI3SEL[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value 0x00000000</td>
<td>Reset value 0x00000000</td>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x05C</td>
<td></td>
<td>0x060</td>
<td>Timx_AF1</td>
<td>0x064</td>
<td>Timx_AF2</td>
<td>0x068</td>
<td>Reserved</td>
<td>0x3DC</td>
<td>Timx_DCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>DBSS[3:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>DBL[4:0]</td>
</tr>
<tr>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x3DC</td>
<td></td>
<td>0x3E0</td>
<td>Timx_DMAR</td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td>DMAB[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td>Reset value 00000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Refer to Section 2.3 for the register boundary addresses.
43 Basic timers (TIM6/TIM7)

43.1 TIM6/TIM7 introduction

The basic timers TIM6/TIM7 consist in a 16-bit autoreload counter driven by a programmable prescaler.
They can be used as generic timers for time-base generation.
The timers are completely independent, and do not share any resources.

43.2 TIM6/TIM7 main features

Basic timer (TIM6/TIM7) features include:
- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Interrupt/DMA generation on the update event: counter overflow.

43.3 TIM6/TIM7 functional description

43.3.1 TIM6/TIM7 block diagram

Figure 568. Basic timer block diagram

Notes:
- Reg Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA
43.3.2 TIM6/TIM7 internal signals

The table in this section summarizes the TIM inputs and outputs.

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)</td>
</tr>
<tr>
<td>tim_trgo</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_upd_it</td>
<td>Output</td>
<td>Timer update event interrupt</td>
</tr>
<tr>
<td>tim_upd_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
</tbody>
</table>

43.3.3 TIM6/TIM7 clocks

The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

*Figure 569* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

*Figure 569. Control circuit in normal mode, internal clock divided by 1*
43.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-Reload register (TIMx_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 570 and Figure 571 give some examples of the counter behavior when the prescaler ratio is changed on the fly.
**Figure 570. Counter timing diagram with prescaler division change from 1 to 2**

<table>
<thead>
<tr>
<th>tim_psc_ck</th>
<th>CEN</th>
<th>tim_cnt_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Counter register: F7  F8  F9  FA  FB  FC  00  01  02  03

Update event (UEV)

Prescaler control register: 0  1

Write a new value in TIMx_PSC

Prescaler buffer: 0  1

Prescaler counter: 0  0  1  0  1  0  1

**Figure 571. Counter timing diagram with prescaler division change from 1 to 4**

<table>
<thead>
<tr>
<th>tim_psc_ck</th>
<th>CEN</th>
<th>tim_cnt_ck</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Counter register: F7  F8  F9  FA  FB  FC  00  01

Update event (UEV)

Prescaler control register: 0  3

Write a new value in TIMx_PSC

Prescaler buffer: 0  3

Prescaler counter: 0  0  1  2  3  0  1  2  3
43.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

**Figure 572. Counter timing diagram, internal clock divided by 1**
Figure 573. Counter timing diagram, internal clock divided by 2

Figure 574. Counter timing diagram, internal clock divided by 4
Figure 575. Counter timing diagram, internal clock divided by N

Figure 576. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger.

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.
Figure 578 presents the dithering principle applied to four consecutive counting periods.

**Figure 578. Dithering principle**

When the dithering mode is enabled, the register coding is changed as follows (see Figure 579 for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

**Note:** The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

**Figure 579. Data format and register coding in dithering mode**

The minimum frequency is given by the following formula:
Resolution = \( \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{PWMmin}} = \frac{F_{\text{Tim}}}{\text{MaxResolution}} \)

Dithering mode disabled: \( F_{\text{PWMmin}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode enabled: \( F_{\text{PWMmin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \)

Note: The maximum TIMx_ARR value is limited to 0xFFFFE in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on Figure 580, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 580. FCnt resolution vs frequency

The period changes are spread over 16 consecutive periods, as described in Figure 581.

Figure 581. PWM dithering pattern

The autoreload and compare values increments are spread following the specific patterns described in Table 443. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.
43.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register’s bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

### Table 443. TIMx_ARR register change dithering pattern

<table>
<thead>
<tr>
<th>LSB value</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>1110</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
</tr>
<tr>
<td>1111</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>-</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
<td>+1</td>
</tr>
</tbody>
</table>
43.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

*Note:* The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

43.3.8 TIM6/TIM7 DMA requests

The TIM6/TIM7 can generate a single DMA request, as shown in Table 444.

<table>
<thead>
<tr>
<th>DMA acronym</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
</tbody>
</table>

43.3.9 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

43.3.10 TIM6/TIM7 low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

43.3.11 TIM6/TIM7 interrupts

The TIM6/TIM7 can generate a single interrupt, as shown in Table 446.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM6, TIM7</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
43.4 TIM6/TIM7 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

43.4.1 TIMx control register 1 (TIMx_CR1)(x = 6 to 7)

Address offset: 0x00
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
  0: Dithering disabled
  1: Dithering enabled
  
  *Note:* The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
  0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
  1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
  0: TIMx_ARR register is not buffered.
  1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
  0: Counter is not stopped at update event
  1: Counter stops counting at the next update event (clearing the CEN bit).
Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt or DMA request if enabled.
These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value
(ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
43.4.2 TIMx control register 2 (TIMx_CR2)\((x = 6 \text{ to } 7)\)

Address offset: 0x04
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 6:4 MMS[2:0]: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

- **000:** Reset - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).
- **001:** Enable - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.
- **010:** Update - the update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

**Note:** The clock of the slave timer or the peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

43.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)\((x = 6 \text{ to } 7)\)

Address offset: 0x0C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 UDE: Update DMA request enable
- 0: Update DMA request disabled.
- 1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 UIE: Update interrupt enable
- 0: Update interrupt disabled.
- 1: Update interrupt enabled.
### 43.4.4 TIMx status register (TIMx_SR)(x = 6 to 7)

Address offset: 0x10  
Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

- **Bit 0** **UIF**: Update interrupt flag  
  - This bit is set by hardware on an update event. It is cleared by software.
  - 0: No update occurred.
  - 1: Update interrupt pending. This bit is set by hardware when the registers are updated:  
    - On counter overflow if UDIS = 0 in the TIMx_CR1 register.
    - When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

### 43.4.5 TIMx event generation register (TIMx_EGR)(x = 6 to 7)

Address offset: 0x14  
Reset value: 0x0000

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

Bits 15:1 Reserved, must be kept at reset value.

- **Bit 0** **UG**: Update generation  
  - This bit can be set by software, it is automatically cleared by hardware.
  - 0: No action.
  - 1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

### 43.4.6 TIMx counter (TIMx_CNT)(x = 6 to 7)

Address offset: 0x24  
Reset value: 0x00000000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| CNT[15:0] |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

* Non-dithering mode (DITHEN = 0)
  The register holds the counter value.

* Dithering mode (DITHEN = 1)
  The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

### 43.4.7 TIMx prescaler (TIMx_PSC)(x = 6 to 7)

**Address offset**: 0x28

**Reset value**: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency \( f_{\text{tim_cnt_ck}} \) is equal to \( f_{\text{tim_psc_ck}} / (\text{PSC[15:0] + 1}) \).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

### 43.4.8 TIMx autoreload register (TIMx_ARR)(x = 6 to 7)

**Address offset**: 0x2C

**Reset value**: 0x0000 FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to **Section 43.3.4: Time-base unit** for more details about ARR update and behavior.

* Non-dithering mode (DITHEN = 0)
  The counter is blocked while the auto-reload value is null.

* Dithering mode (DITHEN = 1)
  The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.
### 43.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x00   | TIMx_CR1      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x04   | TIMx_CR2      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   | 0   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x08   | Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x0C   | TIMx_DIER     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x10   | TIMx_SR       |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x14   | TIMx_EGR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x18-  | Reserved      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x20   |             |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x24   | TIMx_CNT      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x28   | TIMx_PSC      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x2C   | TIMx_ARR      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Refer to Section 2.3 for the register boundary addresses.
44 General-purpose timers (TIM9/TIM12/TIM13/TIM14)

44.1 TIM9/TIM12/TIM13/TIM14 introduction

The TIM9/TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM9/TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in Section 44.4.20: Timer synchronization (TIM9/TIM12 only).

44.2 TIM9/TIM12 main features

The features of the TIM9/TIM12 general-purpose timers include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare
44.3 TIM13/TIM14 main features

The features of general-purpose timers TIM13/TIM14 include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare
44.4 TIM9/TIM12/TIM13/TIM14 functional description

44.4.1 Block diagram

Figure 582. General-purpose timer block diagram (TIM9/TIM12)

Notes:
- Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt
**Figure 583. General-purpose timer block diagram (TIM13/TIM14)**

1. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See Section 44.4.21: Using timer output as trigger for other timers (TIM13/TIM14 only) for details.

### 44.4.2 TIM9/TIM12/TIM13/TIM14 pins and internal signals

Table 448 and Table 449 in this section summarize the TIM inputs and outputs.

#### Table 448. TIM input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.</td>
</tr>
<tr>
<td>TIM_CH2</td>
<td>Input/Output</td>
<td></td>
</tr>
</tbody>
</table>

1. Available for TIM9/TIM12 only.

#### Table 449. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_ti2_in[15:0]</td>
<td>Input</td>
<td></td>
</tr>
<tr>
<td>tim_itr[15:0]</td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_oc1</td>
<td>Output</td>
<td>Internal timer output. Can be used for triggering other timers or the ADC(s).</td>
</tr>
<tr>
<td>tim_oc2</td>
<td>Output</td>
<td></td>
</tr>
</tbody>
</table>
Table 449. TIM internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_trgo(1)</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3, ..., 16 (maximum value)</td>
</tr>
<tr>
<td>tim_it</td>
<td>Output</td>
<td>Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests</td>
</tr>
</tbody>
</table>

1. Available for TIM9/TIM12 only.

Table 450 and Table 450 list the sources connected to the tim_ti[2:1] input multiplexers.

Table 450. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM9</td>
</tr>
<tr>
<td>tim_ti1_in0</td>
<td>TIM9_CH1</td>
</tr>
<tr>
<td>tim_ti1_in1</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_ti1_in2</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_ti1_in3</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_ti1_in4</td>
<td>MCO1</td>
</tr>
<tr>
<td>tim_ti1_in5</td>
<td>MCO2</td>
</tr>
<tr>
<td>tim_ti1_in[15:6]</td>
<td></td>
</tr>
</tbody>
</table>

Table 451. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti2 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM9</td>
</tr>
<tr>
<td>tim_ti2_in0</td>
<td>TIM9_CH2</td>
</tr>
<tr>
<td>tim_ti2_in[15:1]</td>
<td></td>
</tr>
</tbody>
</table>

Table 452 lists the internal sources connected to the tim_itr input multiplexer.

Table 452. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>TIMx</th>
<th>TIM9</th>
<th>TIM12</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>tim1_trgo</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>tim2_trgo</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
<td>tim4_trgo</td>
</tr>
</tbody>
</table>
### 44.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

#### Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 584* and *Figure 585* give some examples of the counter behavior when the prescaler ratio is changed on the fly.
Figure 584. Counter timing diagram with prescaler division change from 1 to 2

Figure 585. Counter timing diagram with prescaler division change from 1 to 4
44.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM9/TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

**Figure 586. Counter timing diagram, internal clock divided by 1**

![Counter timing diagram]

MSv50997V1
**Figure 587. Counter timing diagram, internal clock divided by 2**

- `tim_psc_ck`
- CEN
- `tim_cnt_ck`

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0034</th>
<th>0035</th>
<th>0036</th>
<th>0000</th>
<th>0001</th>
<th>0002</th>
<th>0003</th>
</tr>
</thead>
</table>

- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)

**Figure 588. Counter timing diagram, internal clock divided by 4**

- `tim_psc_ck`
- CEN
- `tim_cnt_ck`

<table>
<thead>
<tr>
<th>Counter register</th>
<th>0035</th>
<th>0036</th>
<th>0000</th>
<th>0001</th>
</tr>
</thead>
</table>

- Counter overflow
- Update event (UEV)
- Update interrupt flag (UIF)
Figure 589. Counter timing diagram, internal clock divided by N

Figure 590. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)
44.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- **Internal clock (tim_ker_ck)**
- **External clock mode1** (for TIM9/TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- **Internal trigger inputs (tim_itrx)** (for TIM9/TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to Section: Using one timer as prescaler for another timer for more details.

**Internal clock source (tim_ker_ck)**

The internal clock source is the default clock source for TIM13/TIM14.

For TIM9/TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

*Figure 592* shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.
Figure 592. Control circuit in normal mode, internal clock divided by 1

External clock source mode 1 (TIM9/TIM12 only)

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 593. tim_ti2 external clock connection example

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:
1. Select the proper \textit{tim\_ti2\_in}[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx\_TISEL register.

2. Configure channel 2 to detect rising edges on the \textit{tim\_ti2} input by writing CC2S = 01 in the TIMx\_CCMR1 register.

3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx\_CCMR1 register (if no filter is needed, keep IC2F = 0000).

4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx\_CCER register.

5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx\_SMCR register.

6. Select \textit{tim\_ti2} as the trigger input source by writing TS = 110 in the TIMx\_SMCR register.

7. Enable the counter by writing CEN = 1 in the TIMx\_CR1 register.

\textit{Note: The capture prescaler is not used for triggering, it is not necessary to configure it.}

When a rising edge occurs on \textit{tim\_ti2}, the counter counts once and the TIF flag is set.

The delay between the rising edge on \textit{tim\_ti2} and the actual clock of the counter is due to the resynchronization circuit on \textit{tim\_ti2} input.

**Figure 594. Control circuit in external clock mode 1**

### 44.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

\textit{Figure 595, Figure 596, Figure 597} and \textit{Figure 598} give an overview of a capture/compare channel.

The input stage samples the corresponding \textit{tim\_tix} input to generate a filtered signal \textit{tim\_tixf}. Then, an edge detector with polarity selection generates a signal (\textit{tim\_tixfp}) which
can be used as trigger input by the slave mode controller or as the capture command. It is
prescaled before the capture register (ICxPS).

Figure 595. Capture/compare channel 1 input stage (TIM13/TIM14)

Figure 596. Capture/compare channel 1 input stage (TIM9/TIM12)

The output stage generates an intermediate waveform which is then used for reference:
tim_ocxref (active high). The polarity acts at the end of the chain.
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

### 44.4.7 Input capture mode

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be
cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the tim_tix inputs). Let’s imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:
• The TIMx_CCR1 register gets the value of the counter on the active transition.
• CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
• An interrupt is generated depending on the CC1E bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

44.4.8 PWM input mode (TIM9/TIM12 only)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:
• The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
• The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two `tim_icx` signals are mapped on the same `tim_tix` input.
- These two `tim_icx` signals are active on edges with opposite polarity.
- One of the two `tim_tixfpy` signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper `tim_ti1_in[15:0]` source (internal or external) with the `TI1SEL[3:0]` bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the `CC1S` bits to 01 in the TIMx_CCMR1 register (`tim_ti1` selected).
3. Select the active polarity for `tim_ti1fp1` (used both for capture in TIMx_CCR1 and counter clear): program the `CC1P` and `CC1NP` bits to 00 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the `CC2S` bits to 10 in the TIMx_CCMR1 register (`tim_ti1` selected).
5. Select the active polarity for `tim_ti1fp2` (used for capture in TIMx_CCR2): program the `CC2P` and `CC2NP` bits to 10 (active on falling edge).
6. Select the valid trigger input: write the `TS` bits to 00101 in the TIMx_SMCR register (`tim_ti1fp1` selected).
7. Configure the slave mode controller in reset mode: write the `SMS` bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the `CC1E` and `CC2E` bits to 1 in the TIMx_CCER register.

![PWM input mode timing](image)

**Figure 599. PWM input mode timing**

### 44.4.9 Forced output mode

In output mode (`CCxS` bits = 00 in the TIMx_CCMRx register), each output compare signal (`tim_ocxref` and then `tim_ocx`) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.
To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

### 44.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCxIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 600.
44.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per TIM_OCx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

The TIM_OCx polarity is software programmable using the CCxE bit in the TIMx_CCER register. It can be programmed as active high or active low. The TIM_OCx output is enabled by the CCxP bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal TIM_OCxRef is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then TIM_OCxRef is held at 1. If the compare value is 0 then TIM_OCxRef is held at 0. Figure 601 shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.
Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. Figure 602 presents the dithering principle applied to four consecutive PWM cycles.
When the dithering mode is enabled, the register coding is changed as follows (see Figure 603 for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

**Note:** The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

**Figure 603. Data format and register coding in dithering mode**

The minimum frequency is given by the following formula:

\[
\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\max_{\text{Resolution}}}
\]
The maximum TIMx_ARR and TIMxCC Ry values are limited to 0xFFFFF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on Figure 604, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

**Figure 604. PWM resolution vs frequency**

The duty cycle and/or period changes are spread over 16 consecutive periods, as described in Figure 605.
Figure 605. PWM dithering pattern

The autoreload and compare values increments are spread following specific patterns described in Table 453. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 453. CCR and ARR register change dithering pattern

<table>
<thead>
<tr>
<th>-</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSB value</td>
<td>1</td>
</tr>
<tr>
<td>0000</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
</tr>
</tbody>
</table>
44.4.12 Combined PWM mode (TIM9/TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

**Note:** The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

*Figure 606* represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
44.4.13 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

\[ CNT < CCRx \leq ARR \] (in particular, \( 0 < CCRx \))
Figure 607. Example of one pulse mode

For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).
- The tDELAY is defined by the value written in the TIMx_CCR1 register.
- The tPULSE is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over
from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

**Particular case: tim_ocx fast enable**

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay $t_{\text{DELAY min}}$ that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

### 44.4.14 Retriggerable one pulse mode (TIM9/TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in Section 44.4.13: One-pulse mode:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

*Note:* The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

---

**Figure 608. Retriggerable one pulse mode**

<table>
<thead>
<tr>
<th>tim_trgi</th>
<th>Counter</th>
<th>tim_ocx</th>
</tr>
</thead>
</table>

MSv62349v2
44.4.15 **UIF bit remapping**

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

44.4.16 **Timer input XOR function**

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_t1 and tim_t2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in Figure 609.

![Figure 609. Measuring time interval between edges on 2 signals](M5v63068V1)

44.4.17 **TIM9/TIM12 external trigger synchronization**

The TIM9/TIM12 timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_t1 input:

1. Configure the channel 1 to detect rising edges on tim_t1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the
1. Configure the channel 1 to detect low levels on TIM_t1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select TIM_t1 as the input source by writing TS = 00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TIM_t1 is low and stops as soon as TIM_t1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TIM_t1 and the actual stop of the counter is due to the resynchronization circuit on TIM_t1 input.

**Slave mode: Gated mode**

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when TIM_t1 input is low:

1. Configure the channel 1 to detect low levels on TIM_t1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select TIM_t1 as the input source by writing TS = 00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as TIM_t1 is low and stops as soon as TIM_t1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on TIM_t1 and the actual stop of the counter is due to the resynchronization circuit on TIM_t1 input.
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on `tim_ti2` input:

1. Configure the channel 2 to detect rising edges on `tim_ti2`. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps `IC2F = 0000`). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, `CC2S = 01` in TIMx_CCMR1 register. Program `CC2P = 1` and `CC2NP = 0` in TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing `SMS = 110` in TIMx_SMCR register. Select `tim_ti2` as the input source by writing `TS = 00110` in TIMx_SMCR register.

When a rising edge occurs on `tim_ti2`, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on `tim_ti2` and the actual start of the counter is due to the resynchronization circuit on `tim_ti2` input.
44.4.18 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

44.4.19 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
44.4.20 Timer synchronization (TIM9/TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 42.4.23: Timer synchronization for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

44.4.21 Using timer output as trigger for other timers (TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.
44.4.22 ADC triggers (TIM9/TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

44.4.23 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the debug section.

44.5 TIM9/TIM12/TIM13/TIM14 low-power modes

Table 454. Effect of low-power modes on TIM9/TIM12/TIM13/TIM14

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

44.6 TIM9/TIM12/TIM13/TIM14 interrupts

The TIM9/TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in Table 455.

Table 455. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Capture/compare 1</td>
<td></td>
<td>CC1IF</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>TIM</td>
<td>Capture/compare 2(1)</td>
<td>CC2IF</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Trigger(1)</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Available for TIM9/TIM12 only.
44.7 TIM9/TIM12 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

44.7.1 TIMx control register 1 (TIMx_CR1)(x = 9, 12)

Address offset: 0x000

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),

00: tDTS = tim_ker_ck
01: tDTS = 2 × tim_ker_ck
10: tDTS = 4 × tim_ker_ck
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).
Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generates an update interrupt if enabled. These events can be:
– Counter overflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update event (UEV) generation.
0: UEV enabled. An UEV is generated by one of the following events:
– Counter overflow
– Setting the UG bit

Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
CEN is cleared automatically in one-pulse mode, when an update event occurs.
*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 44.7.2 TIM12 control register 2 (TIMx_CR2)(x = 9, 12)

Address offset: 0x004
Reset value: 0x0000
RM0477 General-purpose timers (TIM9/TIM12/TIM13/TIM14)

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **T11S**: tim_ti1 selection
- **0**: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
- **1**: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection
These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:
- **000**: Reset - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.
- **001**: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).
- **010**: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.
- **011**: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).
- **100**: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).
- **101**: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

### 44.7.3 TIMx slave mode control register (TIMx_SMCR)(x = 9, 12)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>ts[4:3]</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>ts[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>sm[3]</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>sm[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>ts[4:3]</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>ts[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>sm[3]</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>sm[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>ts[4:3]</td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>ts[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>sm[3]</td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>sm[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>ts[4:3]</td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>ts[2:0]</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>sm[3]</td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>sm[2:0]</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM**: Master/Slave mode
- **0**: No action
- **1**: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]: Trigger selection**

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

- 00000: Internal Trigger 0 (tim_itr0)
- 00001: Internal Trigger 1 (tim_itr1)
- 00010: Internal Trigger 2 (tim_itr2)
- 00011: Internal Trigger 3 (tim_itr3)
- 00100: tim_ti1 Edge Detector (tim_ti1f_ed)
- 00101: Filtered Timer Input 1 (tim_ti1fp1)
- 00110: Filtered Timer Input 2 (tim_ti2fp2)
- 01000: Internal Trigger 4 (tim_itr4)
- 01001: Internal Trigger 5 (tim_itr5)
- 01010: Internal Trigger 6 (tim_itr6)
- 01011: Internal Trigger 7 (tim_itr7)
- 01100: Internal Trigger 8 (tim_itr8)
- 01101: Internal Trigger 9 (tim_itr9)
- 01110: Internal Trigger 10 (tim_itr10)
- 01111: Internal Trigger 10 (tim_itr11)
- 10000: Internal Trigger 10 (tim_itr12)
- 10001: Internal Trigger 10 (tim_itr13)
- 10010: Internal Trigger 10 (tim_itr14)
- 10011: Internal Trigger 10 (tim_itr15)
- Others: Reserved

See **Table 452: TIMx internal trigger connection** for more details on the meaning of tim_itrx for each timer.

**Note:** These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3  **Reserved**, must be kept at reset value.
Bits 16, 2, 1, 0  **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.
0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.
0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.
1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

**Note:** The gated mode (including gated + reset mode) must not be used if tim_tif_ed is selected as the trigger input (TS = 00100). Indeed, tim_tif_ed outputs 1 pulse for each transition on tim_tif, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

### 44.7.4 TIMx interrupt enable register (TIMx_DIER)(x = 9, 12)

Address offset: 0x00C

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 6  **TIE:** Trigger interrupt enable

0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bits 5:3  Reserved, must be kept at reset value.
Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable  
0: CC2 interrupt disabled.  
1: CC2 interrupt enabled.  

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable  
0: CC1 interrupt disabled.  
1: CC1 interrupt enabled.  

Bit 0 **UIE**: Update interrupt enable  
0: Update interrupt disabled.  
1: Update interrupt enabled.  

### 44.7.5 TIMx status register (TIMx_SR)(x = 9, 12)

Address offset: 0x010  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td><strong>CC2OF</strong>: Capture/compare 2 overcapture flag refer to CC1OF description</td>
</tr>
</tbody>
</table>
| 9     | **CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.  
0: No overcapture has been detected.  
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set  
| 8-7   | Reserved, must be kept at reset value.                                      |
| 6     | **TIF**: Trigger interrupt flag  
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.  
0: No trigger event occurred.  
1: Trigger interrupt pending.  
| 5-3   | Reserved, must be kept at reset value.                                      |
Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag  
 refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag  
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).  
0: No compare match / No input capture occurred  
1: A compare match or an input capture occurred. 

**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description. 

**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on TIM_IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCR1).

Bit 0 **UIF**: Update interrupt flag  
This bit is set by hardware on an update event. It is cleared by software.  
0: No update occurred.  
1: Update interrupt pending. This bit is set by hardware when the registers are updated:  
  – At overflow and if UDIS = 0 in the TIMx_CR1 register.  
  – When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.  
  – When CNT is reinitialized by a trigger event (refer to Section 44.7.3: TIMx slave mode control register (TIMx_SMCR)(x = 9, 12)), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

44.7.6 **TIMx event generation register (TIMx_EGR)(x = 9, 12)**  
Address offset: 0x014  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bit 6 **TG**: Trigger generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3  
Reserved, must be kept at reset value.
44.7.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 9, 12)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
```

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

*Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).*
44.7.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 9, 12)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.
Output compare mode:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bits 31:25: Reserved, must be kept at reset value.
- Bits 23:17: Reserved, must be kept at reset value.
- Bit 15: Reserved, must be kept at reset value.
- Bits 24, 14:12: **OC2M[3:0]**: Output compare 2 mode
  - Refer to **OC1M[3:0]** for bit description.
- Bit 11: **OC2PE**: Output compare 2 preload enable
- Bit 10: **OC2FE**: Output compare 2 fast enable
- Bits 9:8: **CC2S[1:0]**: Capture/Compare 2 selection
  - This bitfield defines the direction of the channel (input/output) as well as the used input.
    - 00: CC2 channel is configured as output
    - 01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
    - 10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
    - 11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
  - **Note**: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).
- Bit 7: Reserved, must be kept at reset value.
Bits 16, 6:4 \textbf{OC1M}[3:0]: Output compare 1 mode (refer to bit 16 for \textbf{OC1M}[3])

These bits define the behavior of the output reference signal \textit{tim\_oc1ref} from which \textit{tim\_oc1} is derived. \textit{tim\_oc1ref} is active high whereas the active level of \textit{tim\_oc1} depends on the \textit{CC1P}.

0000: Frozen - The comparison between the output compare register TIM\textsubscript{x} CCR1 and the counter TIM\textsubscript{x} CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The \textit{tim\_oc1ref} signal is forced high when the TIM\textsubscript{x} CNT counter matches the capture/compare register 1 (TIM\textsubscript{x} CCR1).

0010: Set channel 1 to inactive level on match. The \textit{tim\_oc1ref} signal is forced low when the TIM\textsubscript{x} CNT counter matches the capture/compare register 1 (TIM\textsubscript{x} CCR1).

0011: Toggle - \textit{tim\_oc1ref} toggles when TIM\textsubscript{x} CNT = TIM\textsubscript{x} CCR1

0100: Force inactive level - \textit{tim\_oc1ref} is forced low

0101: Force active level - \textit{tim\_oc1ref} is forced high

0110: PWM mode 1 - channel 1 is active as long as TIM\textsubscript{x} CNT<TIM\textsubscript{x} CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIM\textsubscript{x} CNT<TIM\textsubscript{x} CCR1 else it is active

1000: Retriggable OPM mode 1 - The channel is active until a trigger event is detected (on \textit{tim\_trgi} signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retriggable OPM mode 2 - The channel is inactive until a trigger event is detected (on \textit{tim\_trgi} signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - \textit{tim\_oc1ref} has the same behavior as in PWM mode 1. \textit{tim\_oc1refc} is the logical OR between \textit{tim\_oc1ref} and \textit{tim\_oc2ref}.

1101: Combined PWM mode 2 - \textit{tim\_oc1ref} has the same behavior as in PWM mode 2. \textit{tim\_oc1refc} is the logical AND between \textit{tim\_oc1ref} and \textit{tim\_oc2ref}.

1110: Reserved,

1111: Reserved

\textbf{Note:} In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.
Bit 3 **OC1PE**: Output compare 1 preload enable
- 0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately
- 1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable
This bit is used to accelerate the effect of an event on the trigger in input on the CC output.
- 0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles
- 1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

### 44.7.9 TIMx capture/compare enable register (TIMx_CCER)(x = 9, 12)

Address offset: 0x020
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:10</th>
<th>Bit 9:8</th>
<th>Bit 7:6</th>
<th>Bit 5:4</th>
<th>Bit 3:2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>rw</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity
CC1 channel configured as output: CC1NP must be kept cleared
CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).
Bit 2  Reserved, must be kept at reset value.

Bit 1  **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When **CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of tim_t1fp1 and tim_t2fp1 for trigger or capture operations.

- **CC1NP = 0, CC1P = 0**: non-inverted/rising edge. The circuit is sensitive to tim_t1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_t1fp1 is not inverted (trigger operation in gated mode).
- **CC1NP = 0, CC1P = 1**: inverted/falling edge. The circuit is sensitive to tim_t1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_t1fp1 is inverted (trigger operation in gated mode).
- **CC1NP = 1, CC1P = 1**: non-inverted/both edges/ The circuit is sensitive to both tim_t1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_t1fp1 is not inverted (trigger operation in gated mode).
- **CC1NP = 1, CC1P = 0**: This configuration is reserved, it must not be used.

Bit 0  **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 456. Output control bit for standard tim_ocx channels

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output disabled (not driven by the timer: Hi-Z)</td>
</tr>
<tr>
<td>1</td>
<td>Output enabled (tim_ocx = tim_ocxref + Polarity)</td>
</tr>
</tbody>
</table>

**Note:** The states of the external I/O pins connected to the standard tim_ocx channels depend on the state of the tim_ocx channel and on the GPIO registers.

**44.7.10 TIMx counter (TIMx_CNT)(x = 9, 12)**

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**CNT[15:0]**

<table>
<thead>
<tr>
<th>Bit</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
44.7.11 TIMx prescaler (TIMx_PSC)(x = 9, 12)

Address offset: 0x028
Reset value: 0x0000

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

- **Non-dithering mode (DITHEN = 0)**
  The register holds the counter value.

- **Dithering mode (DITHEN = 1)**
  The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

44.7.12 TIMx autoreload register (TIMx_ARR)(x = 9, 12)

Address offset: 0x02C
Reset value: 0x0000 FFFF

Bits 31:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the Section 44.4.3: Time-base unit for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.

- **Non-dithering mode (DITHEN = 0)**
  The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

- **Dithering mode (DITHEN = 1)**
  The register holds the integer part in ARR[15:4]. The ARR[3:0] bitfield contains the dithered part.
### 44.7.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 9, 12)

Address offset: 0x034  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19-0</td>
<td>CCR1[19:0]: Capture/compare 1 value</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as output:</strong></td>
</tr>
<tr>
<td></td>
<td>CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.</td>
</tr>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as input:</strong></td>
</tr>
<tr>
<td></td>
<td>CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.</td>
</tr>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-12</td>
<td>CCR1[15:12]</td>
</tr>
<tr>
<td>11-8</td>
<td>CCR1[11:8]</td>
</tr>
<tr>
<td>7-4</td>
<td>CCR1[7:4]</td>
</tr>
<tr>
<td>3-0</td>
<td>CCR1[3:0]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19-0</td>
<td>CCR2[19:0]: Capture/compare 2 value</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as output:</strong></td>
</tr>
<tr>
<td></td>
<td>CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs. The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.</td>
</tr>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as input:</strong></td>
</tr>
<tr>
<td></td>
<td>CR2 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR2 register is read-only and cannot be programmed.</td>
</tr>
<tr>
<td></td>
<td>Non-dithering mode (DITHEN = 0)</td>
</tr>
<tr>
<td></td>
<td>The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.</td>
</tr>
<tr>
<td></td>
<td>Dithering mode (DITHEN = 1)</td>
</tr>
<tr>
<td></td>
<td>The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-12</td>
<td>CCR2[15:12]</td>
</tr>
<tr>
<td>11-8</td>
<td>CCR2[11:8]</td>
</tr>
<tr>
<td>7-4</td>
<td>CCR2[7:4]</td>
</tr>
<tr>
<td>3-0</td>
<td>CCR2[3:0]</td>
</tr>
</tbody>
</table>
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR2[19:0]**: Capture/compare 2 value

- **If channel CC2 is configured as output:**
  - CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).
  - It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.
  - The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on TIM_OC2 output.

  - **Non-dithering mode (DITHEN = 0)**
    - The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.
  - **Dithering mode (DITHEN = 1)**
    - The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

- **If channel CC2 is configured as input:**
  - CCR2 is the counter value transferred by the last input capture 1 event (TIM_IC2). The TIMx_CCR2 register is read-only and cannot be programmed.

  - **Non-dithering mode (DITHEN = 0)**
    - The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.
  - **Dithering mode (DITHEN = 1)**
    - The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

### 44.7.15 TIMx timer input selection register (TIMx_TISEL)(x = 9, 12)

**Address offset:** 0x05C

**Reset value:** 0x0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>TI2SEL[3:0]</th>
<th></th>
<th></th>
<th></th>
<th>TI1SEL[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 11:8  **TI2SEL[3:0]**: selects TIM_TI2_IN[15:0] input

- **0000**: TIMx_CH2 input (TIM_TI2_IN0)
- **0001**: TIM_TI2_IN1
- **...**
- **0100**: TIM_TI2_IN15

Refer to *Table 451: Interconnect to the TIM TI2 input multiplexer* for interconnects list.

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  **TI1SEL[3:0]**: selects TIM_TI1_IN[15:0] input

- **0000**: TIMx_CH1 input (TIM_TI1_IN0)
- **0001**: TIM_TI1_IN1
- **...**
- **1111**: TIM_TI1_IN15

Refer to *Table 450: Interconnect to the TIM TI1 input multiplexer* for interconnects list.
## 44.7.16 TIM9/TIM12 register map

TIM9/TIM12 registers are mapped as 16-bit addressable registers as described below:

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | TIMx_CR1      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | TIMx_CR2      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | TIMx_SMCR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | TIMx_DIER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | TIMx_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | TIMx_EGR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x018  | TIMx_CCMR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Input Capture mode |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x01C  | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x020  | TIMx_CCER     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | TIMx_CNT      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | CNT[15:0]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | TIMx_PSC      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PSC[15:0]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.

| Offset   | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x02C    | TIMx_ARR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x030    | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x034    | TIMx_CCR1     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x038    | TIMx_CCR2     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x03C to | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x058    | Res.          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x05C    | TIMx_TISEL    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x060-   | Reserved      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x03E8   | Res.          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 457. TIMx register map and reset values (x = 9, 12) (continued)
44.8 TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

44.8.1 TIMx control register 1 (TIMx_CR1)(x = 13, 14)

Address offset: 0x000
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled

*Note: The DITHEN bit can only be modified when CEN bit is reset.*

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: tDTS = tim_ker_ck
01: tDTS = 2 × tim_ker_ck
10: tDTS = 4 × tim_ker_ck
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).
Bit 2  **URS**: Update request source
This bit is set and cleared by software to select the update interrupt (UEV) sources.
0: Any of the following events generate an UEV if enabled:
   - Counter overflow
   - Setting the UG bit
1: Only counter overflow generates an UEV if enabled.

Bit 1  **UDIS**: Update disable
This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.
0: UEV enabled. An UEV is generated by one of the following events:
   - Counter overflow
   - Setting the UG bit.
   Buffered registers are then loaded with their preload values.
1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0  **CEN**: Counter enable
0: Counter disabled
1: Counter enabled

*Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.*

### 44.8.2  TIMx interrupt enable register (TIMx_DIER)(x = 13, 14)
Address offset: 0x00C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:2</th>
<th>Bit 11:8</th>
<th>Bit 7:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:2</th>
<th>Bit 11:8</th>
<th>Bit 7:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC1IE</td>
<td>UIE</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

### 44.8.3  TIMx status register (TIMx_SR)(x = 13, 14)
Address offset: 0x010
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:2</th>
<th>Bit 11:8</th>
<th>Bit 7:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:2</th>
<th>Bit 11:8</th>
<th>Bit 7:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC1OF</td>
<td>UIF</td>
<td>rc_w0</td>
<td>rc_w0</td>
</tr>
</tbody>
</table>
### Bits 15:10
Reserved, must be kept at reset value.

#### Bit 9 CC1OF
Capture/Compare 1 overcapture flag
- This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
- 0: No overcapture has been detected.
- 1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

#### Bits 8:2
Reserved, must be kept at reset value.

#### Bit 1 CC1IF
Capture/compare 1 interrupt flag
- This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
- 0: No compare match / No input capture occurred
- 1: A compare match or an input capture occurred.

**If channel CC1 is configured as output:** this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

**If channel CC1 is configured as input:** this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on TIM_IN1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

#### Bit 0 UIF
Update interrupt flag
- This bit is set by hardware on an update event. It is cleared by software.
- 0: No update occurred.
- 1: Update interrupt pending. This bit is set by hardware when the registers are updated:
  - At overflow and if UDIS = 0 in the TIMx_CR1 register.
  - When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

## 44.8.4 TIMx event generation register (TIMx_EGR)(x = 13, 14)

**Address offset:** 0x014

**Reset value:** 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CC1G</td>
<td>UG</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

![ST Logo](https://example.com/st_logo.png)

RM0477 Rev 6 1955/3791
Bits 15:2  Reserved, must be kept at reset value.

Bit 1  **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
   - **If channel CC1 is configured as output**: CC1IF flag is set, Corresponding interrupt or is sent if enabled.
   - **If channel CC1 is configured as input**: The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0  **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

### 44.8.5 TIMx capture/compare mode register 1
**(TIMx_CCMR1)(x = 13, 14)**

Address offset: 0x018
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

**Input capture mode:**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.
**44.8.6 TIMx capture/compare mode register 1 [alternate]
(TIMx_CCMR1)(x = 13, 14)**

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and output mode.
Output compare mode:

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>OC1M[2:0]</td>
<td>OC1PE</td>
<td>OC1FE</td>
<td>CC1S[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

Bits 16, 6, 4 **OC1M[3:0]:** Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

**Note:** In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.
44.8.7 TIMx capture/compare enable register
(TIMx_CCER)(x = 13, 14)

Address offset: 0x020
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.
CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).
General-purpose timers (TIM9/TIM12/TIM13/TIM14)

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When **CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of tim_t1f1p1 for capture operations.

- CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_t1f1p1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_t1f1p1 is not inverted (trigger operation in gated mode).
- CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_t1f1p1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_t1f1p1 is inverted (trigger operation in gated mode).
- CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_t1f1p1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_t1f1p1 not inverted (trigger operation in gated mode).
- CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

**Table 458. Output control bit for standard tim_ocx channels**

<table>
<thead>
<tr>
<th>CCxE bit</th>
<th>tim_ocx output state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Output Disabled (tim_ocx = 0)</td>
</tr>
<tr>
<td>1</td>
<td>tim_ocx = tim_ocxref + Polarity</td>
</tr>
</tbody>
</table>

**Note**: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

### 44.8.8 TIMx counter (TIMx_CNT) (x = 13, 14)

Address offset: 0x024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

1960/3791 RM0477 Rev 6
Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

- **Non-dithering mode (DITHEN = 0)**
  - The register holds the counter value.
- **Dithering mode (DITHEN = 1)**
  - The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

### 44.8.9 TIMx prescaler (TIMx_PSC)(x = 13, 14)

**Address offset**: 0x028  
**Reset value**: 0x0000

![PSC Register](image)

Bits 15:0 **PSC[15:0]**: Prescaler value

- The counter clock frequency \( f_{\text{tim_cnt_ck}} \) is equal to \( f_{\text{tim_psc_ck}} / (\text{PSC}[15:0] + 1) \).
- PSC contains the value to be loaded in the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

### 44.8.10 TIMx autoreload register (TIMx_ARR)(x = 13, 14)

**Address offset**: 0x02C  
**Reset value**: 0x0000 FFFF

![ARR Register](image)

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

- **ARR[19:0]** is the value to be loaded in the actual auto-reload register.
- Refer to the **Section 44.4.3: Time-base unit** for more details about ARR update and behavior.
- The counter is blocked while the auto-reload value is null.

- **Non-dithering mode (DITHEN = 0)**
  - The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
- **Dithering mode (DITHEN = 1)**
44.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 13, 14)

Address offset: 0x034
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 CCR1[19:0]: Capture/compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:
CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

44.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 13, 14)

Address offset: 0x05C
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>
44.8.13 **TIM13/TIM14 register map**

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

<table>
<thead>
<tr>
<th>Table 459. TIM13/TIM14 register map and reset values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Offset</strong></td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>0x000</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0x004 to 0x008</td>
</tr>
<tr>
<td>0x00C</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0x010</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0x018</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0x01C</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0x020</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0x024</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **T1SEL[3:0]** selects `tim_ti1_in[15:0]` input

0000: TIMx_Ch1 input (`tim_ti1_in0`)

0001: `tim_ti1_in1`

... 1111: `tim_ti1_in15`

Refer to Table 450: Interconnect to the `tim_ti1` input multiplexer for interconnects list.
Table 459. TIM13/TIM14 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x038  to 0x058</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060 - 0x3E8</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
45 General purpose timers (TIM15/TIM16/TIM17)

45.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a
programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input
signals (input capture) or generating output waveforms (output compare, PWM,
complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several
milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any
resources. TIM15 can be synchronized as described in Section 45.4.26: Timer
synchronization (TIM15 only).

45.2 TIM15 main features

TIM15 includes the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also "on the fly") the counter clock
  frequency by any factor between 1 and 65535
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect
  several timers together
- Repetition counter to update the timer registers only after a given number of cycles of
  the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external
    trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external
    trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
45.3 TIM16/TIM17 main features

The TIM16/TIM17 timers include the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input
45.4 TIM15/TIM16/TIM17 functional description

45.4.1 Block diagram

Figure 613. TIM15 block diagram

Notes:
- Reg: Preload registers transferred to active registers on U event according to control bit
- Event
- Interrupt & DMA output

1. Refer to Section 45.4.15: Using the break function for details.
1. Refer to Section 45.4.15: Using the break function for details.

2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See Section 45.4.27: Using timer output as trigger for other timers (TIM16/TIM17 only) for details.

### 45.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 460 and Table 461 in this section summarize the TIM inputs and outputs.

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_CH1</td>
<td>Input/Output</td>
<td>Timer multi-purpose channels. Each channel can be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.</td>
</tr>
<tr>
<td>TIM_CH2(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIM_CH1N</td>
<td>Output</td>
<td>Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.</td>
</tr>
<tr>
<td>TIM_BKIN</td>
<td>Input / Output</td>
<td>Break input. This input can also be configured in bidirectional mode.</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.
Table 461. TIM internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_ti1_in[15:0]</td>
<td>Input</td>
<td>Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_ti2_in[15:0]</td>
<td>Input</td>
<td>Internal trigger input bus. These inputs can be used for the slave mode controller or as an input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_itr[15:0]</td>
<td>Input</td>
<td>Input Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).</td>
</tr>
<tr>
<td>tim_trgo</td>
<td>Output</td>
<td>Internal trigger output. This trigger can trigger other on-chip peripherals.</td>
</tr>
<tr>
<td>tim_ocref_clr[7:0]</td>
<td>Input</td>
<td>Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulselength control.</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td>Input</td>
<td>Break input for internal signals</td>
</tr>
<tr>
<td>tim_sys_brk[n:0]</td>
<td>Input</td>
<td>System break input. This input gathers the MCU's system level errors.</td>
</tr>
<tr>
<td>tim_pclk</td>
<td>Input</td>
<td>Timer APB clock</td>
</tr>
<tr>
<td>tim_ker_ck</td>
<td>Input</td>
<td>Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)</td>
</tr>
<tr>
<td>tim_it</td>
<td>Output</td>
<td>Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Output</td>
<td>Timer capture / compare 1 dma request</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Output</td>
<td>Timer capture / compare 2 dma request</td>
</tr>
<tr>
<td>tim_upd_dma</td>
<td>Output</td>
<td>Timer update dma request</td>
</tr>
<tr>
<td>tim_trg_dma</td>
<td>Output</td>
<td>Timer trigger dma request</td>
</tr>
<tr>
<td>tim_com_dma</td>
<td>Output</td>
<td>Timer commutation dma request</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.

Table 462 and Table 463 list the sources connected to the tim_ti[2:1] input multiplexers.

Table 462. Interconnect to the tim_ti1 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM15</td>
</tr>
<tr>
<td>tim_ti1_in0</td>
<td>TIM15_CH1</td>
</tr>
</tbody>
</table>
Table 462. Interconnect to the tim_ti1 input multiplexer (continued)

<table>
<thead>
<tr>
<th>tim_ti1 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM15</td>
</tr>
<tr>
<td>tim_ti1_in1</td>
<td>TIM2_CH1</td>
</tr>
<tr>
<td>tim_ti1_in2</td>
<td>TIM3_CH1</td>
</tr>
<tr>
<td>tim_ti1_in3</td>
<td>TIM4_CH1</td>
</tr>
<tr>
<td>tim_ti1_in4</td>
<td>MCO1</td>
</tr>
<tr>
<td>tim_ti1_in5</td>
<td>MCO2</td>
</tr>
<tr>
<td>tim_ti1_in[15:6]</td>
<td></td>
</tr>
</tbody>
</table>

Table 463. Interconnect to the tim_ti2 input multiplexer

<table>
<thead>
<tr>
<th>tim_ti2 inputs</th>
<th>Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TIM15</td>
</tr>
<tr>
<td>tim_ti2_in0</td>
<td>TIM15.CH2</td>
</tr>
<tr>
<td>tim_ti2_in1</td>
<td>TIM2.CH2</td>
</tr>
<tr>
<td>tim_ti2_in2</td>
<td>TIM3.CH2</td>
</tr>
<tr>
<td>tim_ti2_in3</td>
<td>TIM4.CH2</td>
</tr>
<tr>
<td>tim_ti2_in[15:4]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 464 lists the internal sources connected to the tim_itr input multiplexer.

Table 464. TIMx internal trigger connection

<table>
<thead>
<tr>
<th>tim_itr inputs</th>
<th>TIM15</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_itr0</td>
<td>tim1_trgo</td>
</tr>
<tr>
<td>tim_itr1</td>
<td>tim2_trgo</td>
</tr>
<tr>
<td>tim_itr2</td>
<td>tim3_trgo</td>
</tr>
<tr>
<td>tim_itr3</td>
<td>tim4_trgo</td>
</tr>
<tr>
<td>tim_itr4</td>
<td>tim5_trgo</td>
</tr>
<tr>
<td>tim_itr5</td>
<td>tim9_trgo</td>
</tr>
<tr>
<td>tim_itr6</td>
<td>tim12_trgo</td>
</tr>
<tr>
<td>tim_itr7</td>
<td>tim13_oc1</td>
</tr>
<tr>
<td>tim_itr8</td>
<td>tim14_oc1</td>
</tr>
<tr>
<td>tim_itr9</td>
<td>Reserved</td>
</tr>
<tr>
<td>tim_itr10</td>
<td>tim16_oc1</td>
</tr>
<tr>
<td>tim_itr11</td>
<td>tim17_oc1</td>
</tr>
<tr>
<td>tim_itr[15:12]</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 465 and Table 466 list the sources connected to the tim_brk input.

### Table 465. Timer break interconnect

<table>
<thead>
<tr>
<th>tim_brk inputs</th>
<th>TIM1</th>
<th>TIM16</th>
<th>TIM17</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM_BKIN</td>
<td>TIM15_BKIN pin</td>
<td>TIM16_BKIN pin</td>
<td>TIM17_BKIN pin</td>
</tr>
<tr>
<td>tim_brk_cmp[8:1]</td>
<td></td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

### Table 466. System break interconnect

<table>
<thead>
<tr>
<th>tim_sys_brk inputs</th>
<th>TIM1/TIM8</th>
<th>Enable bit in SBS_BRK_LOCKUPR register</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim Sys brk0</td>
<td>AXISRAM1 double ECC error</td>
<td>ARAM1ECC_BL</td>
</tr>
<tr>
<td>tim Sys brk1</td>
<td>AXISRAM3 double ECC error</td>
<td>ARAM3ECC_BL</td>
</tr>
<tr>
<td>tim Sys brk2</td>
<td>ITCM double ECC error</td>
<td>ITCMECC_BL</td>
</tr>
<tr>
<td>tim Sys brk3</td>
<td>DTCM double ECC error</td>
<td>DTCMECC_BL</td>
</tr>
<tr>
<td>tim Sys brk4</td>
<td>Backup RAM double ECC error</td>
<td>BKRA MECC_BL</td>
</tr>
<tr>
<td>tim Sys brk5</td>
<td>Cortex-M7 LOCKUP</td>
<td>CM7LCKUP_BL</td>
</tr>
<tr>
<td>tim Sys brk6</td>
<td>FLASH double ECC error</td>
<td>FLASHECC_BL</td>
</tr>
<tr>
<td>tim Sys brk7</td>
<td>Programmable Voltage Detector (PVD)</td>
<td>PVD_BL</td>
</tr>
<tr>
<td>CSS</td>
<td>Clock Security System</td>
<td>None (always enabled)</td>
</tr>
</tbody>
</table>

### 45.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:
- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).
Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

**Prescaler description**

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

*Figure 615* and *Figure 616* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

**Figure 615. Counter timing diagram with prescaler division change from 1 to 2**

![Counter timing diagram with prescaler division change from 1 to 2](image)
### 45.4.4 Counter modes

#### Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.
When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

**Figure 617. Counter timing diagram, internal clock divided by 1**

![Diagram of counter timing](image-url)
Figure 618. Counter timing diagram, internal clock divided by 2

Figure 619. Counter timing diagram, internal clock divided by 4
Figure 620. Counter timing diagram, internal clock divided by N

Figure 621. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)
45.4.5 Repetition counter

Section 45.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to Figure 623). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.
**45.4.6 Clock selection**

The counter clock can be provided by the following clock sources:

- **Internal clock (tim_ker_ck)**
- **External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)**
- **Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer to enable another timer* for more details.**

**Internal clock source (tim_ker_ck)**

If the slave mode controller is disabled (SMS = 000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed.
only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 624 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

**Figure 624. Control circuit in normal mode, internal clock divided by 1**

![Control circuit in normal mode, internal clock divided by 1](image)

**External clock source mode 1**

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

**Figure 625. tim_ti2 external clock connection example**

![tim_ti2 external clock connection example](image)

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.
The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to
the resynchronization circuit on tim_ti2 input.

Figure 626. Control circuit in external clock mode 1

45.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a
shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and
an output stage (with comparator and output control).

Figure 627 to Figure 630 give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal
tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which
can be used as trigger input by the slave mode controller or as the capture command. It is
prescaled before the capture register (ICxPS).
The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 627. Capture/compare channel (example: channel 1 input stage)

Figure 628. Capture/compare channel 1 main circuit
The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

45.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was
already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_t1 input rises. To do this, use the following procedure:

1. Select the proper tim_t1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.

2. Select the active input: TIMx_CCR1 must be linked to the tim_t1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.

3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let’s imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_t1 when eight consecutive samples with the new level have been detected (sampled at fDTS frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

4. Select the edge of the active transition on the tim_t1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).

5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).

6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.

7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:
- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.
45.4.9 **PWM input mode (only for TIM15)**

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single `tim_tix` input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two `tim_icx` signals are mapped on the same `tim_tix` input.
- These two `tim_icx` signals are active on edges with opposite polarity.
- One of the two `tim_tixfp` signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on `tim_t1` using the following procedure (depending on TIM_ker_ck frequency and prescaler value):

1. Select the proper `tim_t1`[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (`tim_t1` selected).
3. Select the active polarity for `tim_t1fp1` (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (`tim_t1` selected).
5. Select the active polarity for `tim_t1fp2` (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (`tim_t1fp1` selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.
45.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

45.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set
active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in Figure 632.
45.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to Upcounting mode on page 1973.
In the following example applies to PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. Figure 633 shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

**Figure 633. Edge-aligned PWM waveforms (ARR = 8)**

| Counter register | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 1 |
| tim_ocxref       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCRx = 4         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| tim_ocxref       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCRx = 7         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCxIF            |CMS=01|CMS=10|CMS=11|
| tim_ocxref       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCRx = 8         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCxIF            |CMS=01|CMS=10|CMS=11|
| tim_ocxref       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCRx > 8         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCxIF            |CMS=01|CMS=10|CMS=11|
| tim_ocxref       |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCRx = 0         |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| CCxIF            |CMS=01|CMS=10|CMS=11|

**Dithering mode**

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The Figure 634 below presents the dithering principle applied to four consecutive PWM cycles.
When the dithering mode is enabled, the register coding is changed as follows (see Figure 635 for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{\text{Tim}}}{F_{\text{pwm}}} \Rightarrow F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{\text{MaxResolution}}$$
Dithering mode disabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65536} \)

Dithering mode enabled: \( F_{\text{pwmMin}} = \frac{F_{\text{Tim}}}{65535 + \frac{15}{16}} \)

Note: The maximum TIMx.ARR and TIMx.CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

As shown on the Figure 636 below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

**Figure 636. PWM resolution vs frequency**

The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the Figure 637 below.
The autoreload and compare values increments are spread following specific patterns described in the Table 467 below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

**Table 467. CCR and ARR register change dithering pattern**

<table>
<thead>
<tr>
<th>-</th>
<th>PWM period</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LSB value</strong></td>
<td>1</td>
</tr>
<tr>
<td>0000</td>
<td>-</td>
</tr>
<tr>
<td>0001</td>
<td>+1</td>
</tr>
<tr>
<td>0010</td>
<td>+1</td>
</tr>
<tr>
<td>0011</td>
<td>+1</td>
</tr>
<tr>
<td>0100</td>
<td>+1</td>
</tr>
<tr>
<td>0101</td>
<td>+1</td>
</tr>
<tr>
<td>0110</td>
<td>+1</td>
</tr>
<tr>
<td>0111</td>
<td>+1</td>
</tr>
<tr>
<td>1000</td>
<td>+1</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
</tr>
<tr>
<td>1010</td>
<td>+1</td>
</tr>
<tr>
<td>1011</td>
<td>+1</td>
</tr>
<tr>
<td>1100</td>
<td>+1</td>
</tr>
<tr>
<td>1101</td>
<td>+1</td>
</tr>
</tbody>
</table>
45.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

**Note:** The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

*Figure 638* represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
45.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to Table 474: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 2054 for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 639. Complementary output with symmetrical dead-time insertion.

![Diagram](MSv62332V1)

The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on Figure 640.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

**Note:** If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.
The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to Section 45.8.14: TIMx break and dead-time register (TIMx_BDTR) (x = 16 to 17) on page 2058 for delay calculation.
Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

*Note:* When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

45.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shutdown level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to Table 474: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) on page 2054 for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.
The break is generated by the tim_brk inputs which have:
- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:
- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per Figure 643 below.

**Figure 643. Break circuitry overview**

Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.
When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable outputs remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to Section 45.8.14: TIMx break and dead-time register (TIMx_BDTR) (x = 16 to 17). The LOCK bits can be written only once after an MCU reset.

The Figure 644 shows an example of behavior for the outputs in response to a break.
Figure 644. Output behavior in response to a break event on tim_brk

- tim_ocxref:
- tim_ocx (tim_ocxn not implemented, CCxP=0, OISx=1)
- tim_ocx (tim_ocxn not implemented, CCxP=0, OISx=0)
- tim_ocx (tim_ocxn not implemented, CCxP=1, OISx=1)
- tim_ocx (tim_ocxn not implemented, CCxP=1, OISx=0)
- tim_ocx
- tim_ocxn (CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=0, OISxN=1)
- tim_ocxn (CCxE=1, CCxP=0, OISx=1, CCxNE=1, CCxNP=1, OISxN=1)
- tim_ocxn (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1)
- tim_ocxn (CCxE=1, CCxP=0, OISx=0, CCxNE=0, CCxNP=0, OISxN=1 or OISx=OISxN=0)
- tim_ocxn (CCxE=1, CCxP=0, CCxNE=0, CCxNP=0, OISx=0 or OISx=OISxN=1)
45.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on Figure 645.

They are used to have:
- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMxBDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:
- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see Table 468).

<table>
<thead>
<tr>
<th>MOE</th>
<th>BKBID</th>
<th>BKDSRM</th>
<th>Break protection state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Armed</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Disarmed</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>Armed</td>
</tr>
</tbody>
</table>

Arming and rearming break circuitry

The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).
The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

45.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on Figure 646 below.
45.4.18  6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The Figure 647 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.
Figure 647. 6-step generation, COM example (OSSR = 1)

Example 1

Counter (CNT) (CCRx)

tim_ocxref

Write COM to 1

CCxE = 1
CCxNE = 0
OCxM = 0010 (forced inactive)

Write OCxM to 0100

CCxE = 1
CCxNE = 0
OCxM = 0100

Example 2

tim_ocx

Write CCxNE to 1 and OCxM to 0101

CCxE = 1
CCxNE = 0
OCxM = 0100 (forced inactive)

CCxE = 0
CCxNE = 1
OCxM = 0101

Example 3

tim_ocx

Write CCxNE to 0 and OCxM to 0100

CCxE = 1
CCxNE = 0
OCxM = 0010 (forced inactive)

CCxE = 1
CCxNE = 1
OCxM = 0100

Counter (CNT)

tim_ocxref

COM event

Example 1

tim_ocx

tim_ocxn

Example 2

tim_ocx

tim_ocxn

Example 3

tim_ocx

tim_ocxn
45.4.19 One-pulse mode

One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- $\text{CNT} < \text{CCR}_x \leq \text{ARR}$ (in particular, $0 < \text{CCR}_x$).

For example one may want to generate a positive pulse on $\text{tim}_{\text{oc}1}$ with a length of $t_{\text{PULSE}}$ and after a delay of $t_{\text{DELAY}}$ as soon as a positive edge is detected on the $\text{tim}_{\text{ti}2}$ input pin.

Let’s use $\text{tim}_{\text{ti}2\text{fp}2}$ as trigger 1:

1. Select the proper $\text{tim}_{\text{ti}2\text{in}[15:1]}$ source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map $\text{tim}_{\text{ti}2\text{fp}2}$ to $\text{tim}_{\text{ti}2}$ by writing CC2S = 01 in the TIMx_CCMR1 register.
3. $\text{tim}_{\text{ti}2\text{fp}2}$ must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure $\text{tim}_{\text{ti}2\text{fp}2}$ as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. $\text{tim}_{\text{ti}2\text{fp}2}$ is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).
The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The \( t_{\text{DELAY}} \) is defined by the value written in the TIMx_CCR1 register.
- The \( t_{\text{PULSE}} \) is defined by the difference between the autoreload value and the compare value \( (\text{TIMx\_ARR} - \text{TIMx\_CCR1}) \).
- Let’s say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

**Particular case: tim_ocx fast enable**

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay \( t_{\text{DELAY\ min}} \) that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

**45.4.20 Retriggerable one pulse mode (TIM15 only)**

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in Section 45.4.19:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

**Note:** The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

*This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.*
45.4.21 **UIF bit remapping**

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

45.4.22 **Timer input XOR function (TIM15 only)**

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 650*.

45.4.23 **External trigger synchronization (TIM15 only)**

The TIM timers are linked together internally for timer synchronization or chaining.
The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

**Slave mode: Reset mode**

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIMx_CCRER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

![Control circuit in reset mode](MSv62361V1)
Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 652. Control circuit in gated mode

![Diagram of control circuit in gated mode](MSv62362V1)
Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).

2. Configure the timer in trigger mode by writing SMS = 110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

![Figure 653. Control circuit in trigger mode](MSv62363V1)

Slave mode selection preload for run-time update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

45.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

45.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).
45.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to Section 42.4.23: Timer synchronization for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the TIM_TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

45.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination’s timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

45.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the TIM_TRGO signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

45.4.29 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see Section 45.8.19: TIMx DMA control register (TIMx_DCR)(x = 16 to 17) for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows:
   - DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6.

Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

*Note:* A null value can be written to the reserved registers.

### 45.4.30 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in Table 469.

<table>
<thead>
<tr>
<th>DMA request signal</th>
<th>DMA request</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tim_upd_dma</td>
<td>Update</td>
<td>UDE</td>
</tr>
<tr>
<td>tim_cc1_dma</td>
<td>Capture/compare 1</td>
<td>CC1DE</td>
</tr>
<tr>
<td>tim_cc2_dma</td>
<td>Capture/compare 2</td>
<td>CC2DE</td>
</tr>
<tr>
<td>tim_com_dma¹</td>
<td>Commutation (COM)</td>
<td>COMDE</td>
</tr>
<tr>
<td>tim_trg_dma¹</td>
<td>Trigger</td>
<td>TDE</td>
</tr>
</tbody>
</table>

¹ Available for TIM15 only.
45.4.31 Debug mode

When the microcontroller enters debug mode (Cortex®-M7 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the debug section.

45.5 TIM15/TIM16/TIM17 low-power modes

Table 470. Effect of low-power modes on TIM15/TIM16/TIM17

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The timer operation is stopped and the register content is kept. No interrupt can be generated.</td>
</tr>
<tr>
<td>Standby</td>
<td>The timer is powered-down and must be reinitialized after exiting the Standby mode.</td>
</tr>
</tbody>
</table>

45.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in Table 471.

Table 471. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop and Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIM</td>
<td>Update</td>
<td>UIF</td>
<td>UIE</td>
<td>write 0 in UIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 1</td>
<td>CC1F</td>
<td>CC1IE</td>
<td>write 0 in CC1IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Capture/compare 2(1)</td>
<td>CC2F</td>
<td>CC2IE</td>
<td>write 0 in CC2IF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Commutation (COM)</td>
<td>COMIF</td>
<td>COMIE</td>
<td>write 0 in COMIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Trigger(1)</td>
<td>TIF</td>
<td>TIE</td>
<td>write 0 in TIF</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Break</td>
<td>BIF</td>
<td>BIE</td>
<td>write 0 in BIF</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

1. Available for TIM15 only.
45.7 TIM15 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

45.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>DITH</td>
<td>UIFRE</td>
<td>MAP</td>
<td>CKD[1:0]</td>
<td>ARPE</td>
<td>OPM</td>
<td>URS</td>
<td>UDIS</td>
<td>CEN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled

*Note:* The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 UIFREMAP: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 CKD[1:0]: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_tix)
00: tDTS = tim_ker_ck
01: tDTS = 2*tim_ker_ck
10: tDTS = 4*tim_ker_ck
11: Reserved

Bit 7 ARPE: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 OPM: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)
Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

- 0: Any of the following events generate an update interrupt if enabled. These events can be:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
- 1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

- 0: UEV enabled. The Update (UEV) event is generated by one of the following events:
  - Counter overflow/underflow
  - Setting the UG bit
  - Update generation through the slave mode controller
  Buffered registers are then loaded with their preload values.
- 1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

- 0: Counter disabled
- 1: Counter enabled

**Note:** External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

---

### 45.7.2 TIM15 control register 2 (TIM15_CR2)

**Address offset:** 0x04

**Reset value:** 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)

- 0: tim_oc2 = 0 when MOE = 0
- 1: tim_oc2 = 1 when MOE = 0

**Note:** This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

- 0: tim_oc1n = 0 after a dead-time when MOE = 0
- 1: tim_oc1n = 1 after a dead-time when MOE = 0

**Note:** This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

- 0: tim_oc1 = 0 after a dead-time when MOE = 0
- 1: tim_oc1 = 1 after a dead-time when MOE = 0

**Note:** This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).
Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers outputs are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.

*Note: This bit acts only on channels that have a complementary output.*

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

*Note: This bit acts only on channels that have a complementary output.*
45.7.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE**: SMS preload enable
This bit selects whether the SMS[3:0] bitfield is preloaded.
0: SMS[3:0] bitfield is not preloaded
1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM**: Master/slave mode
0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.
Bits 21, 20, 6, 5, 4  **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

- **00000:** Internal Trigger 0 (tim_itr0)
- **00001:** Internal Trigger 1 (tim_itr1)
- **00010:** Internal Trigger 2 (tim_itr2)
- **00011:** Internal Trigger 3 (tim_itr3)
- **00100:** tim_ti1 Edge Detector (tim_ti1f_ed)
- **00101:** Filtered Timer Input 1 (tim_ti1fp1)
- **00110:** Filtered Timer Input 2 (tim_ti2fp2)
- **00111:** Reserved
- **01000:** Internal Trigger 4 (tim_itr4)
- **01001:** Internal Trigger 5 (tim_itr5)
- **01010:** Internal Trigger 6 (tim_itr6)
- **01011:** Internal Trigger 7 (tim_itr7)
- **01100:** Internal Trigger 8 (tim_itr8)
- **01101:** Internal Trigger 9 (tim_itr9)
- **01110:** Internal Trigger 10 (tim_itr10)
- **10000:** Internal trigger 12 (tim_itr12)
- **10001:** Internal trigger 13 (tim_itr13)
- **10010:** Internal trigger 14 (tim_itr14)
- **10011:** Internal trigger 15 (tim_itr15)
- **Others:** Reserved

See [Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals](#) for more details on tim_itrx meaning for each timer.

**Note:** These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3  Reserved, must be kept at reset value.
Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.
0001: Reserved
0010: Reserved
0011: Reserved
0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.
0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.
0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.
0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.
1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.
1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

**Note:** The gated mode must not be used if tim_ti1fp1 is selected as the trigger input (TS = 00100). Indeed, tim_ti1fp1 outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

### 45.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

  0: Trigger DMA request disabled
  1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

  0: COM DMA request disabled
  1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.
Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

### 45.7.5 TIM15 status register (TIM15_SR)

Address offset: 0x10
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td>rc_w0</td>
<td></td>
</tr>
</tbody>
</table>

---

RM0477 Rev 6 2019/3791
Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to **CC1OF** description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

- 0: No overcapture has been detected
- 1: The counter value has been captured in TIM15_CCR1 register while **CC1IF** flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

- 0: No break event occurred
- 1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

- 0: No trigger event occurred
- 1: Trigger interrupt pending

Bit 5 **COMIF**: COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –**CCxE**, **CCxNE**, **OCxM**– have been updated). It is cleared by software.

- 0: No COM event occurred
- 1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.
Bit 2 CC2IF: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 CC1IF: Capture/Compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCRER).

Bit 0 UIF: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM15_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS = 0 and UDIS = 0 in the TIM15_CR1 register.
– When CNT is reinitialized by a trigger event (refer to Section 45.7.3: TIM15 slave mode control register (TIM15_SMCR)), if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

### 45.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 BG: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 TG: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled
Bit 5  **COMG**: Capture/Compare control update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits  
*Note*:  This bit acts only on channels that have a complementary output.

Bits 4:3  Reserved, must be kept at reset value.

Bit 2  **CC2G**: Capture/Compare 2 generation  
Refer to **CC1G** description

Bit 1  **CC1G**: Capture/Compare 1 generation  
This bit is set by software in order to generate an event, it is automatically cleared by hardware.  
0: No action  
1: A capture/compare event is generated on channel 1:  
   *If channel CC1 is configured as output:*  
   The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled.  
   *If channel CC1 is configured as input:*  
   The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0  **UG**: Update generation  
This bit can be set by software, it is automatically cleared by hardware.  
0: No action  
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

### 45.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

**Address offset**: 0x18  
**Reset value**: 0x0000 0000  

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Input capture mode**

Bits 31:16  **Reserved**, must be kept at reset value.

Bits 15:12  **IC2F[3:0]**: Input capture 2 filter

Bits 11:10  **IC2PSC[1:0]**: Input capture 2 prescaler
Bits 9:8 **CC2S[1:0]:** Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
- 00: CC2 channel is configured as output
- 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
- 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
- 11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

*Note:* CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bits 7:4 **IC1F[3:0]:** Input capture 1 filter
This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:
- 0000: No filter, sampling is done at fDTS
- 0001: \( f_{SAMPLING} = f_{tim\_ker\_ck}, \ N = 2 \)
- 0010: \( f_{SAMPLING} = f_{tim\_ker\_ck}, \ N = 4 \)
- 0011: \( f_{SAMPLING} = f_{tim\_ker\_ck}, \ N = 8 \)
- 0100: \( f_{SAMPLING} = f_{DTS}/2, \ N = 6 \)
- 0101: \( f_{SAMPLING} = f_{DTS}/2, \ N = 8 \)
- 0110: \( f_{SAMPLING} = f_{DTS}/4, \ N = 6 \)
- 0111: \( f_{SAMPLING} = f_{DTS}/4, \ N = 8 \)
- 1000: \( f_{SAMPLING} = f_{DTS}/8, \ N = 6 \)
- 1001: \( f_{SAMPLING} = f_{DTS}/8, \ N = 8 \)
- 1010: \( f_{SAMPLING} = f_{DTS}/16, \ N = 5 \)
- 1011: \( f_{SAMPLING} = f_{DTS}/16, \ N = 6 \)
- 1100: \( f_{SAMPLING} = f_{DTS}/16, \ N = 8 \)
- 1101: \( f_{SAMPLING} = f_{DTS}/32, \ N = 5 \)
- 1110: \( f_{SAMPLING} = f_{DTS}/32, \ N = 6 \)
- 1111: \( f_{SAMPLING} = f_{DTS}/32, \ N = 8 \)

Bits 3:2 **IC1PSC[1:0]:** Input capture 1 prescaler
This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM15_CCER register).
- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]:** Capture/Compare 1 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
- 00: CC1 channel is configured as output
- 01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
- 10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
- 11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

*Note:* CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).
45.7.8  TIM15 capture/compare mode register 1 [alternate]  
(TIM15_CCMR1)

Address offset: 0x18  
Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

| Bit 31:25 | Reserved, must be kept at reset value. |
| Bit 23:17 | Reserved, must be kept at reset value. |
| Bit 15    | OC2CE: Output compare 2 clear enable |
| Bit 14:12 | OC2M[3:0]: Output compare 2 mode |
| Bit 11    | OC2PE: Output compare 2 preload enable |
| Bit 10    | OC2FE: Output compare 2 fast enable |
| Bit 9:8   | CC2S[1:0]: Capture/Compare 2 selection |
|           | This bitfield defines the direction of the channel (input/output) as well as the used input. |
|           | 00: CC2 channel is configured as output. |
|           | 01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2. |
|           | 10: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register) |
| Note:     | CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER). |
| Bit 7     | OC1CE: Output compare 1 clear enable |
| 0:        | tim_oc1ref is not affected by the tim_ocref_clr_int input. |
| 1:        | tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input. |
### Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

<table>
<thead>
<tr>
<th>OC1M[3:0]</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.</td>
</tr>
<tr>
<td>0001</td>
<td>Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).</td>
</tr>
<tr>
<td>0010</td>
<td>Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).</td>
</tr>
<tr>
<td>0011</td>
<td>Toggle - tim_oc1ref toggles when TIM15_CNT = TIM15_CCR1.</td>
</tr>
<tr>
<td>0100</td>
<td>Force inactive level - tim_oc1ref is forced low.</td>
</tr>
<tr>
<td>0101</td>
<td>Force active level - tim_oc1ref is forced high.</td>
</tr>
<tr>
<td>0110</td>
<td>PWM mode 1 - Channel 1 is active as long as TIM15_CNT&lt;TIM15_CCR1 else inactive.</td>
</tr>
<tr>
<td>0111</td>
<td>PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT&lt;TIM15_CCR1 else active.</td>
</tr>
<tr>
<td>1000</td>
<td>Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.</td>
</tr>
<tr>
<td>1001</td>
<td>Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.</td>
</tr>
<tr>
<td>1010</td>
<td>Reserved</td>
</tr>
<tr>
<td>1011</td>
<td>Reserved</td>
</tr>
<tr>
<td>1100</td>
<td>Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1ref is the logical OR between tim_oc1ref and tim_oc2ref.</td>
</tr>
<tr>
<td>1101</td>
<td>Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1ref is the logical AND between tim_oc1ref and tim_oc2ref.</td>
</tr>
<tr>
<td>1110</td>
<td>Reserved</td>
</tr>
<tr>
<td>1111</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

**Note:** These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.
45.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description
Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

- **CC1 channel configured as output:**
  - 0: `tim_oc1n` active high
  - 1: `tim_oc1n` active low
- **CC1 channel configured as input:**
  This bit is used in conjunction with CC1P to define the polarity of `tim_ti1fp1` and `tim_ti2fp1`. Refer to CC1P description.

  **Note:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output). On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

- 0: Off - `tim_oc1n` is not active. `tim_oc1n` level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
- 1: On - `tim_oc1n` signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

- **CC1 channel configured as output:**
  - 0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
  - 1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)
  
  **When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of T11FP1 and T12FP1 for trigger or capture operations.

  - **CC1NP = 0, CC1P = 0:** non-inverted/rising edge. The circuit is sensitive to T1xFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode).
  - **CC1NP = 0, CC1P = 1:** inverted/falling edge. The circuit is sensitive to T1xFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is inverted (trigger operation in gated mode).
  - **CC1NP = 1, CC1P = 1:** non-inverted/both edges/ The circuit is sensitive to both T1xFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), T1xFP1 is not inverted (trigger operation in gated mode).
  - **CC1NP = 1, CC1P = 0:** this configuration is reserved, it must not be used.

  **Note:** This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register). On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

- 0: Capture mode disabled / OC1 is not active (see below)
- 1: Capture mode enabled / OC1 signal is output on the corresponding output pin

  **When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits, regardless of the CC1E bits state. Refer to **Table 472** for details.
### Table 472. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states</th>
<th>Output states</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
<td>OSSR bit</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| | 0 | 0 | 1 | 0 | Output Disabled (not driven by the timer: Hi-Z) |
| | | | | | tim_ocx = 0 |
| | | | | | tim_ocxref + Polarity |
| | | | | | CCxNP |

| | 0 | 1 | 0 | 1 | Output Disabled (not driven by the timer: Hi-Z) |
| | | | | | tim_ocx = 0 |
| | | | | | tim_ocxref = tim_ocxref XOR |
| | | | | | CCxP |
| | X | 1 | 1 | 1 | Complementary to tim_ocxref (not OCREF) + Polarity + dead-time |
| | | | | | tim_ocxref + Polarity + dead-time |

| | 1 | 0 | 1 | 1 | Off-State (output enabled with inactive state) |
| | | | | | tim_ocx = CCxP |
| | | | | | tim_ocxref + Polarity |
| | | | | | CCxNP |

| | 1 | 1 | 0 | 0 | Off-State (output enabled with inactive state) |
| | | | | | tim_ocx = CCxP |
| | | | | | tim_ocxref = tim_ocxref XOR |
| | | | | | CCxNP |

| 0 | 0 | X | X | 0 | Off-disabled (not driven by the timer: Hi-Z) |
| 0 | 0 | | | | |

| 1 | X | 0 | 1 | 0 | Off-State (output enabled with inactive state) |
| | | | | | Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP |

| 1 | X | 1 | 1 | 1 | |
| | | | | | |

---

**Note:** The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

---

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.
45.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>UIF</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>CNT[15:0]</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>CNT[15:0]</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>CNT[15:0]</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>CNT[15:0]</td>
<td>r</td>
</tr>
</tbody>
</table>

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
- Non-dithering mode (DITHEN = 0)
  The register holds the counter value.
- Dithering mode (DITHEN = 1)
  The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

45.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>PSC[15:0]</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency \(f_{\text{cnt}_{\text{ck}}}\) is equal to \(f_{\text{pasc}_{\text{ck}}} / (PSC[15:0] + 1)\).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).
### 45.7.12 TIM15 autoreload register (TIM15_ARR)

Address offset: 0x2C  
Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>ARR[19:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the Section 45.4.3: Time-base unit on page 1971 for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

### 45.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

- The number of PWM periods in edge-aligned mode.
- The number of half PWM period in center-aligned mode.
### 45.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-20</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19-0</td>
<td>CCR1[19:0]: Capture/compare 1 value</td>
<td></td>
</tr>
</tbody>
</table>

#### If channel CC1 is configured as output:
- CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
- It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
- The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

#### Non-dithering mode (DITHEN = 0)
- The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

#### Dithering mode (DITHEN = 1)
- The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

#### If channel CC1 is configured as input:
- CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

#### Non-dithering mode (DITHEN = 0)
- The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

#### Dithering mode (DITHEN = 1)
- The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.
**45.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)**

Address offset: 0x38  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

**Non-dithering mode (DITHEN = 0)**

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

**Non-dithering mode (DITHEN = 0)**

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

**Dithering mode (DITHEN = 1)**

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

---

**45.7.16 TIM15 break and dead-time register (TIM15_BDTR)**

Address offset: 0x44  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE</td>
<td>AOE</td>
<td>BKP</td>
<td>BKE</td>
<td>OSSR</td>
<td>OSSI</td>
<td>LOCK[1:0]</td>
<td>DTG[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.
Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional
- 0: Break input tim_brk in input mode
- 1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

*Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

*Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm
- 0: Break input tim_brk is armed
- 1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

*Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.*

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter
This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
- 0000: No filter, tim_brk acts asynchronously
- 0001: \( f_{SAMPLING} = f_{tim\_ker\_ck} \), \( N = 2 \)
- 0010: \( f_{SAMPLING} = f_{tim\_ker\_ck} \), \( N = 4 \)
- 0011: \( f_{SAMPLING} = f_{tim\_ker\_ck} \), \( N = 8 \)
- 0100: \( f_{SAMPLING} = f_{DTS}/2 \), \( N = 6 \)
- 0101: \( f_{SAMPLING} = f_{DTS}/2 \), \( N = 8 \)
- 0110: \( f_{SAMPLING} = f_{DTS}/4 \), \( N = 6 \)
- 0111: \( f_{SAMPLING} = f_{DTS}/4 \), \( N = 8 \)
- 1000: \( f_{SAMPLING} = f_{DTS}/8 \), \( N = 6 \)
- 1001: \( f_{SAMPLING} = f_{DTS}/8 \), \( N = 8 \)
- 1010: \( f_{SAMPLING} = f_{DTS}/16 \), \( N = 5 \)
- 1011: \( f_{SAMPLING} = f_{DTS}/16 \), \( N = 6 \)
- 1100: \( f_{SAMPLING} = f_{DTS}/16 \), \( N = 8 \)
- 1101: \( f_{SAMPLING} = f_{DTS}/32 \), \( N = 5 \)
- 1110: \( f_{SAMPLING} = f_{DTS}/32 \), \( N = 6 \)
- 1111: \( f_{SAMPLING} = f_{DTS}/32 \), \( N = 8 \)

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*
Bit 15  **MOE**: Main output enable
This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.
0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.
1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)
See tim_ocx/tim_ocxn enable description for more details (Section 45.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2026).

Bit 14  **AOE**: Automatic output enable
0: MOE can be set only by software
1: MOE can be set by software or automatically at the next update event (if the break input is not be active)
**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13  **BKP**: Break polarity
0: Break input tim_brk is active low
1: Break input tim_brk is active high
**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12  **BKE**: Break enable
0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled
1: Break inputs (tim_brk and tim_sys_brk clock failure event) enabled
This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
**Note:** Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11  **OSSR**: Off-state selection for Run mode
This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.
See tim_ocx/tim_ocxn enable description for more details (Section 45.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2026).
0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)
1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).
**Note:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10  **OSSI**: Off-state selection for Idle mode
This bit is used when MOE = 0 on channels configured as outputs.
See tim_ocx/tim_ocxn enable description for more details (Section 45.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2026).
0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)
1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1, tim_ocx/tim_ocxn enable output signal = 1)
**Note:** This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).
45.7.17 **TIM15 timer deadtime register 2 (TIM15_DTR2)**

Address offset: 0x054

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bit 17 **DTPE**: Deadtime preload enable
0: Deadtime value is not preloaded
1: Deadtime value preload is enabled

Note: *This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).*
General purpose timers (TIM15/TIM16/TIM17) RM0477

45.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 TI2SEL[3:0]: selects tim_ti2_in[15:0] input

0000: TIM15_CH2 input (tim_ti2_in0)
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 TI1SEL[3:0]: selects tim_ti1_in[15:0] input

0000: TIM15_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for interconnects list.

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 DTAE: Deadtime asymmetric enable
0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register
1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 DTGF[7:0]: Dead-time falling edge generator setup
This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x tdtg with tdtg = tDTS:
DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xtdtg with Tdtg = 2xtdtg:
DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xtdtg with Tdtg = 8xtdtg:
DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xtdtg with Tdtg = 16xtdtg:

Example if T_DTS = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 μs to 31750 ns by 250 ns steps,
32 μs to 63 μs by 1 μs steps,
64 μs to 126 μs by 2 μs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).
45.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060
Reset value: 0x0000 0001

Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 BKCMP4P: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 BKCMP3P: tim_brk_cmp3 input polarity
This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp3 input is active high
1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 BKCMP2P: tim_brk_cmp2 input polarity
This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp2 input is active high
1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 BKCMP1P: tim_brk_cmp1 input polarity
This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp1 input is active high
1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
Bit 9 **BKINP**: TIMx_BKIN input polarity
This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: TIMx_BKIN input is active high
1: TIMx_BKIN input is active low

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable
This bit enables the tim_brk_cmp8 for the timer’s tim_brk input. mdf_brkx output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp8 input disabled
1: tim_brk_cmp8 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable
This bit enables the tim_brk_cmp7 for the timer’s tim_brk input. COMP7 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp7 input disabled
1: tim_brk_cmp7 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable
This bit enables the tim_brk_cmp6 for the timer’s tim_brk input. tim_brk_cmp6 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp6 input disabled
1: tim_brk_cmp6 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable
This bit enables the tim_brk_cmp5 for the timer’s tim_brk input. tim_brk_cmp5 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp5 input disabled
1: tim_brk_cmp5 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable
This bit enables the tim_brk_cmp4 for the timer’s tim_brk input. tim_brk_cmp4 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp4 input disabled
1: tim_brk_cmp4 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable
This bit enables the tim_brk_cmp3 for the timer’s tim_brk input. tim_brk_cmp3 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp3 input disabled
1: tim_brk_cmp3 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).*
Bit 2 **BKCMP2E**: tim_brk_cmp2 enable  
This bit enables the tim_brk_cmp2 for the timer’s tim_brk input. tim_brk_cmp2 output is ‘ORed’ with the other tim_brk sources.  
0: tim_brk_cmp2 input disabled  
1: tim_brk_cmp2 input enabled  
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable  
This bit enables the tim_brk_cmp1 for the timer’s tim_brk input. tim_brk_cmp1 output is ‘ORed’ with the other tim_brk sources.  
0: tim_brk_cmp1 input disabled  
1: tim_brk_cmp1 input enabled  
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable  
This bit enables the TIMx_BKIN alternate function input for the timer’s tim_brk input. TIMx_BKIN input is ‘ORed’ with the other tim_brk sources.  
0: TIMx_BKIN input disabled  
1: TIMx_BKIN input enabled  
*Note*: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

### 45.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
| OCRSEL[2:0]: ocref_clr source selection  
These bits select the ocref_clr input source.  
000: tim_ocref_clr0  
001: tim_ocref_clr1  
010: tim_ocref_clr2  
011: tim_ocref_clr3  
100: tim_ocref_clr4  
101: tim_ocref_clr5  
110: tim_ocref_clr6  
111: tim_ocref_clr7  
Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation.  
*Note*: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).
Bits 15:0  Reserved, must be kept at reset value.

**45.7.21  TIM15 DMA control register (TIM15_DCR)**

Address offset: 0x3DC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>DBSS[3:0]</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  **DBSS[3:0]: DMA burst source selection**  
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved  
0001: Update  
0010: CC1  
0011: CC2  
0110: COM  
0111: Trigger  
Other: reserved

Bits 15:13  Reserved, must be kept at reset value.

Bits 12:8  **DBL[4:0]: DMA burst length**  
This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).

00000: 1 transfer,  
00001: 2 transfers,  
00010: 3 transfers,  
...  
10001: 18 transfers.

Bits 7:5  Reserved, must be kept at reset value.

Bits 4:0  **DBA[4:0]: DMA base address**  
This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:  
00000: TIM15_CR1,  
00001: TIM15_CR2,  
00010: TIM15_SMCR,  
...
45.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0
Reset value: 0x0000 0000

Bits 31:0 DMAB[31:0]: DMA register for burst accesses
A read or write operation to the DMAR register accesses the register located at the address (TIM15_CR1 address) + (DBA + DMA index) x 4 where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

45.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

| Offset | Register name | S15 | S14 | S13 | S12 | S11 | S10 | S9 | S8 | S7 | S6 | S5 | S4 | S3 | S2 | S1 | S0 |
|--------|---------------|-----|-----|-----|-----|-----|-----|----|----|----|----|----|----|----|----|----|
| 0x00   | TIM15_CR1     |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | TIM15_CR2     |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | TIM15_SMCR    |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | TIM15_DIER    |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | TIM15_SR      |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x14   | TIM15_EGR     |     |     |     |     |     |     |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0   | 0   | 0   | 0   | 0   | 0   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
0x54

TIM15_DTR2

0x58

2042/3791

0

0

0

0

Reserved

RM0477 Rev 6

Reserved

0

0

OSSI

0

0

0

0

0

0

0

0

Res.

Res.

Res.

LOCK
[1:0]

Res.

0

OSSR

BKF[3:0]

Res.

Reserved

BKE

0

Res.

0

BKP

0
0

AOE

0
0

Res.

0
0
0
1
1
1
1
1
1
1
1

Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.
Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

UIFCPY or Res.

Res.

Res.

OC1M[3]
OC2CE

OC2PE
OC2FE

0
0
0
0
0
0
0
0

Res.
Res.
Res.
Res.
Res.
Res.
Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

OC2M[3]

Res.

Res.

Res.

Res.

Res.

Res.

Res.

CC2S
[1:0]

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

0

MOE

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

Reset value
0

DTAE

Res.

Res.

Res.

0

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Reset value

Res.

Res.

Res.

Res.

Res.

Res.

Res.

BKDSRM

Res.

Res.

Res.

Reset value

DTPE

Reset value

0

Res.

0x48 0x50
Res.

Reset value

Res.

Reset value

Res.

TIM15_BDTR

Res.

0x38 0x40

Res.

TIM15_CCR2

Res.

TIM15_CCR1
Res.

TIM15_RCR

Res.

0x44
TIM15_ARR

BKBID

0x38
TIM15_PSC

Res.

0x34
0

Res.

0x30
Reset value

Res.

0x2C

Res.

0x28
TIM15_CNT

Res.

0x24
TIM15_CCER

Res.

0x20
TIM15_CCMR1
Output
Compare mode

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

Res.

TIM15_CCMR1
Input Capture
mode

Res.

0x18
0

0

0

0

0

Res.

0

0

0

0

0

0

0

0

0

0

0

0

0

0
0
0

Reset value
0

CNT[15:0]

0
0

0
0

0
0

0

0
0
0
0

0

0

0

0

Reset value

0

0

0

0

0

0

CCR1[19:0]

CCR2[19:0]

Res.

DT[7:0]

0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0

0
0
0
0
0
0
0
0

0

0

0

0

0

0

0

CC1E

0

0

CC1P

0

0

CC1NE

0

0

CC2E

0

CC1NP

0

CC2P

0
OC1CE

OC2M
[2:0]

CC2S
[1:0]

Res.

0
IC2
PSC
[1:0]

Res.

IC2F[3:0]

CC2NP

0

0

0

0

OC1M
[2:0]

0

0

IC1
PSC
[1:0]
CC1S
[1:0]

0
0
0

OC1FE

IC1F[3:0]

OC1PE

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

Register
name

Res.

Offset

Res.

General purpose timers (TIM15/TIM16/TIM17)
RM0477

Table 473. TIM15 register map and reset values (continued)

CC1S
[1:0]

0
0
0

0

0

0

0

0
0
0
0
0
0

0
0
0
0
0
0

PSC[15:0]

ARR[19:0]

0
0
0
0
0
0
0
0

1
1
1
1
1
1
1
1

REP[7:0]

DTGF[7:0]


### Table 473. TIM15 register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Refer to Section 2.3 on page 149 for the register boundary addresses.
45.8 TIM16/TIM17 registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions.

45.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15:13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9:8</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6:4</th>
<th>Bit 3</th>
<th>Bit 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>DITHEN</td>
<td>UIFREMAP</td>
<td>CKD[1:0]</td>
<td>ARPE</td>
<td>OPM</td>
<td>URS</td>
<td>UDIS</td>
<td>CEN</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled

*Note:* The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (tDTS) used by the dead-time generators and the digital filters (tim_tix),

00: tDTS = ttim_ker_ck
01: tDTS = 2*ttim_ker_ck
10: tDTS = 4*ttim_ker_ck
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled.
These events can be:
  – Counter overflow/underflow
  – Setting the UG bit
  – Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.
Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
*Note*: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

45.8.2 **TIMx control register 2 (TIMx_CR2)(x = 16 to 17)**
Address offset: 0x04
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 after a dead-time when MOE = 0
1: tim_oc1 = 1 after a dead-time when MOE = 0
*Note*: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).
*Note*: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.
Bit 0 **CCPC**: Capture/compare preloaded control
- 0: CCxE, CCxNE and OCxM bits are not preloaded
- 1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.

*Note: This bit acts only on channels that have a complementary output.*

### 45.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
- 0: CC1 DMA request disabled
- 1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
- 0: Update DMA request disabled
- 1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
- 0: Break interrupt disabled
- 1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE**: COM interrupt enable
- 0: COM interrupt disabled
- 1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
- 0: CC1 interrupt disabled
- 1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
- 0: Update interrupt disabled
- 1: Update interrupt enabled
### 45.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10  
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>0</th>
</tr>
</thead>
</table>

| rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 | rc_w0 |

Bits 15:10 Reserved, must be kept at reset value.

**Bit 9 CC1OF**: Capture/Compare 1 overcapture flag  
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.  
0: No overcapture has been detected  
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

**Bit 8 Reserved, must be kept at reset value.**

**Bit 7 BIF**: Break interrupt flag  
This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.  
0: No break event occurred  
1: An active level has been detected on the break input

**Bit 6 Reserved, must be kept at reset value.**

**Bit 5 COMIF**: COM interrupt flag  
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.  
0: No COM event occurred  
1: COM interrupt pending

**Bits 4:2 Reserved, must be kept at reset value.**

**Bit 1 CC1IF**: Capture/Compare 1 interrupt flag  
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).  
0: No compare match / No input capture occurred  
1: A compare match or an input capture occurred  

**If channel CC1 is configured as output**: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.  

**If channel CC1 is configured as input**: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).
Bit 0 **UIF**: Update interrupt flag
   This bit is set by hardware on an update event. It is cleared by software.
   0: No update occurred.
   1: Update interrupt pending. This bit is set by hardware when the registers are updated:
      – At overflow regarding the repetition counter value (update if repetition counter = 0)
         and if the UDIS = 0 in the TIMx_CR1 register.
      – When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if
         URS = 0 and UDIS = 0 in the TIMx_CR1 register.

45.8.5 **TIMx event generation register (TIMx_EGR)(x = 16 to 17)**

Address offset: 0x14
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>BG: Break generation</td>
</tr>
<tr>
<td></td>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
</tr>
<tr>
<td></td>
<td>0: No action.</td>
</tr>
<tr>
<td></td>
<td>1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>COMG: Capture/Compare control update generation</td>
</tr>
<tr>
<td></td>
<td>This bit can be set by software, it is automatically cleared by hardware.</td>
</tr>
<tr>
<td></td>
<td>0: No action</td>
</tr>
<tr>
<td></td>
<td>1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits</td>
</tr>
<tr>
<td></td>
<td><em>Note</em>: This bit acts only on channels that have a complementary output.</td>
</tr>
<tr>
<td>4:2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>CC1G: Capture/Compare 1 generation</td>
</tr>
<tr>
<td></td>
<td>This bit is set by software in order to generate an event, it is automatically cleared by hardware.</td>
</tr>
<tr>
<td></td>
<td>0: No action.</td>
</tr>
<tr>
<td></td>
<td>1: A capture/compare event is generated on channel 1:</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as output:</strong> CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.</td>
</tr>
<tr>
<td></td>
<td><strong>If channel CC1 is configured as input:</strong> The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.</td>
</tr>
<tr>
<td>0</td>
<td>UG: Update generation</td>
</tr>
<tr>
<td></td>
<td>This bit can be set by software, it is automatically cleared by hardware.</td>
</tr>
<tr>
<td></td>
<td>0: No action.</td>
</tr>
<tr>
<td></td>
<td>1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).</td>
</tr>
</tbody>
</table>
45.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) 
(x = 16 to 17)

Address offset: 0x18
Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

### Input capture mode

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

Bits 7:4: **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_t1 input and the length of the digital filter applied to tim_t1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

- 0000: No filter, sampling is done at fDTS
- 0001: fSAMPLING = ftim_ker_ck, N = 2
- 0010: fSAMPLING = ftim_ker_ck, N = 4
- 0011: fSAMPLING = ftim_ker_ck, N = 8
- 0100: fSAMPLING = fDTS/2, N = 6
- 0101: fSAMPLING = fDTS/2, N = 8
- 0110: fSAMPLING = fDTS/4, N = 6
- 0111: fSAMPLING = fDTS/4, N = 8
- 1000: fSAMPLING = fDTS/8, N = 6
- 1001: fSAMPLING = fDTS/8, N = 8
- 1010: fSAMPLING = fDTS/16, N = 5
- 1011: fSAMPLING = fDTS/16, N = 6
- 1100: fSAMPLING = fDTS/32, N = 5
- 1101: fSAMPLING = fDTS/32, N = 6
- 1110: fSAMPLING = fDTS/32, N = 8
- 1111: fSAMPLING = fDTS/32, N = 8

Bits 3:2: **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

- 00: no prescaler, capture is done each time an edge is detected on the capture input.
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events
Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection
   This bitfield defines the direction of the channel (input/output) as well as the used input.
   00: CC1 channel is configured as output
   01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
   Others: Reserved
   *Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).*

45.8.7 TIMx capture/compare mode register 1 [alternate]
(TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OC1M[2:0]</td>
<td>OC1PE</td>
<td>OC1FE</td>
<td>CC1S[1:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Output compare mode:**

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

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable
   0: tim_oc1ref is not affected by the tim_ocref_clr input.
   1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.
Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).
45.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high
1: tim_oc1n active low
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

*Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).
On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.*
Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity
0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

**When CC1 channel is configured as input**, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

- CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TixFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TixFP1 is not inverted (trigger operation in gated mode).
- CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TixFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TixFP1 is inverted (trigger operation in gated mode).
- CC1NP = 1, CC1P = 1: non-inverted/both edges. The circuit is sensitive to both TixFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TixFP1 is not inverted (trigger operation in gated mode).
- CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

**Note**: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable
0: Capture mode disabled / OC1 is not active (see below)
1: Capture mode enabled / OC1 signal is output on the corresponding output pin

**When CC1 channel is configured as output**, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to Table 474 for details.
### Table 474. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

<table>
<thead>
<tr>
<th>Control bits</th>
<th>Output states(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOE bit</td>
<td>OSSI bit</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

### Note:

The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.
45.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>UIFCPY: UIF Copy</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 30:16 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>CNT[15:0]: Counter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-dithering mode (DITHEN = 0)</td>
<td></td>
</tr>
<tr>
<td>The register holds the counter value.</td>
<td></td>
</tr>
<tr>
<td>Dithering mode (DITHEN = 1)</td>
<td></td>
</tr>
<tr>
<td>The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.</td>
<td></td>
</tr>
</tbody>
</table>

45.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28
Reset value: 0x000000

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>PSC[15:0]: Prescaler value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The counter clock frequency (tim_cnt_ck) is equal to f_{tim_psc_ck} / (PSC[15:0] + 1).</td>
<td></td>
</tr>
<tr>
<td>PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).</td>
<td></td>
</tr>
</tbody>
</table>
45.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C
Reset value: 0x0000 FFFF

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>ARR[15:0]</td>
</tr>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 ARR[19:0]: Auto-reload value
- ARR is the value to be loaded in the actual auto-reload register.
- Refer to the Section 45.4.3: Time-base unit on page 1971 for more details about ARR update and behavior.
- The counter is blocked while the auto-reload value is null.
- Non-dithering mode (DITHEN = 0)
  - The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
- Dithering mode (DITHEN = 1)

45.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30
Reset value: 0x0000

<table>
<thead>
<tr>
<th>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 REP[7:0]: Repetition counter reload value
- This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.
- When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.
- It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:
  - The number of PWM periods in edge-aligned mode.
  - The number of half PWM period in center-aligned mode.
45.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCR1[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20  Reserved, must be kept at reset value.

Bits 19:0  **CCR1[19:0]**: Capture/Compare 1 value

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

*Non-dithering mode (DITHEN = 0)*
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

*Dithering mode (DITHEN = 1)*
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

**If channel CC1 is configured as input:**
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).

*Non-dithering mode (DITHEN = 0)*
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

*Dithering mode (DITHEN = 1)*
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.
45.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44
Reset value: 0x0000 0000

| Bit 31:29 | Reserved, must be kept at reset value. |
| Bit 28   | **BKBID**: Break Bidirectional |
| 0: Break input tim_brk in input mode |
| 1: Break input tim_brk in bidirectional mode |
| In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices. |
| **Note**: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register). Any write operation to this bit takes a delay of 1 APB clock cycle to become effective. |
| Bit 27   | Reserved, must be kept at reset value. |
| Bit 26   | **BKDSRM**: Break Disarm |
| 0: Break input tim_brk is armed |
| 1: Break input tim_brk is disarmed |
| This bit is cleared by hardware when no break source is active. The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared. |
| **Note**: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective. |
Bits 25:20  Reserved, must be kept at reset value.

Bits 19:16  **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

- 0000: No filter, tim_brk acts asynchronously
- 0001: \( f_{SAMPLING} = f_{tim\_ker\_ck} \), \( N = 2 \)
- 0010: \( f_{SAMPLING} = f_{tim\_ker\_ck} \), \( N = 4 \)
- 0011: \( f_{SAMPLING} = f_{tim\_ker\_ck} \), \( N = 8 \)
- 0100: \( f_{SAMPLING} = f_{DTS}/2 \), \( N = 6 \)
- 0101: \( f_{SAMPLING} = f_{DTS}/2 \), \( N = 8 \)
- 0110: \( f_{SAMPLING} = f_{DTS}/4 \), \( N = 6 \)
- 0111: \( f_{SAMPLING} = f_{DTS}/4 \), \( N = 8 \)
- 1000: \( f_{SAMPLING} = f_{DTS}/8 \), \( N = 6 \)
- 1001: \( f_{SAMPLING} = f_{DTS}/8 \), \( N = 8 \)
- 1010: \( f_{SAMPLING} = f_{DTS}/16 \), \( N = 5 \)
- 1011: \( f_{SAMPLING} = f_{DTS}/16 \), \( N = 6 \)
- 1100: \( f_{SAMPLING} = f_{DTS}/32 \), \( N = 5 \)
- 1101: \( f_{SAMPLING} = f_{DTS}/32 \), \( N = 6 \)
- 1110: \( f_{SAMPLING} = f_{DTS}/32 \), \( N = 8 \)

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15  **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

- 0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.
- 1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (Section 45.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 2052).

Bit 14  **AOE**: Automatic output enable

- 0: MOE can be set only by software
- 1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 13  **BKP**: Break polarity

- 0: Break input tim_brk is active low
- 1: Break input tim_brk is active high

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12  **BKE**: Break enable

- 0: Break inputs (tim_brk and tim_sys_brk event) disabled
- 1: Break inputs (tim_brk and tim_sys_brk event) enabled

*Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.
Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (Section 45.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 2052).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)
1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (Section 45.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) on page 2052).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)
1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

*Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).*

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected
01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBD/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.
10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCMRx register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.
11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

*Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.*

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_dtg with t_d tg = t_DTS
DTG[7:5] = 10x → DT = (64+DTG[5:0])x t_d tg with T_d tg = 2x t_DTS
DTG[7:5] = 110 → DT = (32+DTG[4:0])x t_d tg with T_d tg = 8x t_DTS
DTG[7:5] = 111 → DT = (32+DTG[4:0])x t_d tg with T_d tg = 16x t_DTS

Example if T_DTS=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

*Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).*
45.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)\((x = 16 \text{ to } 17)\)

Address offset: 0x054
Reset value: 0x0000 0000

| Bit 31:18 | Reserved, must be kept at reset value. |
| Bit 17   | **DTPE**: Deadtime preload enable |
| 0        | Deadtime value is not preloaded |
| 1        | Deadtime value preload is enabled |
| **Note**: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register). |
| Bit 16   | **DTAE**: Deadtime asymmetric enable |
| 0        | Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register |
| 1        | Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits. |
| **Note**: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register). |
| Bits 15:8 | Reserved, must be kept at reset value. |
| Bits 7:0 | **DTGF[7:0]**: Dead-time falling edge generator setup |
| This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge. |
| DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{d_{tg}} with t_{d_{tg}} = t_{DTS}. |
| DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])x t_{d_{tg}} with T_{d_{tg}} = 2x t_{DTS}. |
| DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])x t_{d_{tg}} with T_{d_{tg}} = 8x t_{DTS}. |
| DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])x t_{d_{tg}} with T_{d_{tg}} = 16x t_{DTS}. |
| Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are: |
| 0 to 15875 ns by 125 ns steps, |
| 16 μs to 31750 ns by 250 ns steps, |
| 32 μs to 63 μs by 1 μs steps, |
| 64 μs to 126 μs by 2 μs steps |
| **Note**: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register). |
### 45.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C  
Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 3:0  **TISEL[3:0]:** selects tim_ti1_in[15:0] input

- 0000: TIMx_CH1 input (tim_ti1_in0)
- 0001: tim_ti1_in1
- ...
- 1111: tim_ti1_in15

Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for interconnects list.

### 45.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060  
Reset value: 0x0000 0001

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14  Reserved, must be kept at reset value.

Bit 13  **BKCMP4P:** tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

- 0: tim_brk_cmp4 input is active high
- 1: tim_brk_cmp4 input is active low

**Note:** This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 12 BKCMP3P: tim_brk_cmp3 input polarity
This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp3 input is active high
1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 BKCMP2P: tim_brk_cmp2 input polarity
This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp2 input is active high
1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 BKCMP1P: tim_brk_cmp1 input polarity
This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp1 input is active high
1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 BKINP: TIMx_BKIN input polarity
This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.
0: TIMx_BKIN input is active high
1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 BKCMP8E: tim_brk_cmp8 enable
This bit enables the tim_brk_cmp8 for the timer’s tim_brk input. mdf_brkx output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp8 input disabled
1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 BKCMP7E: tim_brk_cmp7 enable
This bit enables the tim_brk_cmp7 for the timer’s tim_brk input. tim_brk_cmp7 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp7 input disabled
1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 BKCMP6E: tim_brk_cmp6 enable
This bit enables the tim_brk_cmp6 for the timer’s tim_brk input. tim_brk_cmp6 output is ‘ORed’ with the other tim_brk sources.
0: tim_brk_cmp6 input disabled
1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer’s tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled
1: tim_brk_cmp5 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer’s tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled
1: tim_brk_cmp4 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer’s tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled
1: tim_brk_cmp3 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer’s tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled
1: tim_brk_cmp2 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer’s tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled
1: tim_brk_cmp1 input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer’s tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled
1: TIMx_BKIN input enabled

*Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).*
### 45.8.18 TIMx alternate function register 2 (TIMx_AF2)\( (x = 16\) to 17)\)

Address offset: 0x064  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-19</th>
<th>OW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.  
Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection  
These bits select the tim_ocref_clr input source.  
000: tim_ocref_clr0  
001: tim_ocref_clr1  
010: tim_ocref_clr2  
011: tim_ocref_clr3  
100: tim_ocref_clr4  
101: tim_ocref_clr5  
110: tim_ocref_clr6  
111: tim_ocref_clr7  
Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation.  
*Note:* These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).  
Bits 15:0 Reserved, must be kept at reset value.

### 45.8.19 TIMx DMA control register (TIMx_DCR)\( (x = 16\) to 17)\)

Address offset: 0x3DC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>OW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
<th>RW</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.  
Bits 15:0 **DBSS[3:0]**: tim_ocref_clr source selection  
These bits select the tim_ocref_clr input source.  
000: tim_ocref_clr0  
001: tim_ocref_clr1  
010: tim_ocref_clr2  
011: tim_ocref_clr3  
100: tim_ocref_clr4  
101: tim_ocref_clr5  
110: tim_ocref_clr6  
111: tim_ocref_clr7  
Refer to Section 45.4.2: TIM15/TIM16/TIM17 pins and internal signals for product specific implementation.  
*Note:* These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).  
Bits 15:0 Reserved, must be kept at reset value.
45.8.20 TIM16/TIM17 DMA address for full transfer
(TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0
Reset value: 0x0000 0000

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]:** DMA burst source selection
This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).
- 0000: Reserved
- 0001: Update
- 0010: CC1
- Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]:** DMA burst length
This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).
- 00000: 1 transfer,
- 00001: 2 transfers,
- 00010: 3 transfers,
- ...
- 10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]:** DMA base address
This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.
Example:
- 00000: TIMx_CR1,
- 00001: TIMx_CR2,
- 00010: TIMx_SMCR,
- ...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.
Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

\[(\text{TIM}_x\_\text{CR1 address}) + (\text{DBA} + \text{DMA index}) \times 4\]

where TIM$_x$\_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM$_x$\_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM$_x$\_DCR).
### 45.8.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

#### Table 475. TIM16/TIM17 register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  | Offset | Register name |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---
Refer to Section 2.3 for the register boundary addresses.
46 Low-power timer (LPTIM)

46.1 Introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. Also, the LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

46.2 LPTIM main features

- 16 bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by Pulse Counter application)
- 16 bit ARR autoreload register
- 16 bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable Digital Glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture
46.3 LPTIM implementation

Table 476 describes LPTIM implementation on STM32H7Rx/7Sx devices. The full set of features is implemented in LPTIM1, LPTIM2 and LPTIM3. LPTIM4 and LPTIM5 support a smaller set of features.

Table 476. STM32H7Rx/7Sx LPTIM features

<table>
<thead>
<tr>
<th>LPTIM modes/features(1)</th>
<th>LPTIM1</th>
<th>LPTIM2</th>
<th>LPTIM3</th>
<th>LPTIM4</th>
<th>LPTIM5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PWM mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Input Capture</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Number of channels</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Number of DMA requests</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wake-up in Stop mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X = supported.
46.4 LPTIM functional description

46.4.1 LPTIM block diagram

Figure 654. LPTIM1/2/3 timer block diagram

1. Some IOs may not be available, refer to Section 46.4.2: LPTIM pins and internal signals.
46.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 477. LPTIM1/2/3 input/output pins

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_IN1</td>
<td>Digital input</td>
<td>LPTIM Input 1 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_IN2</td>
<td>Digital input</td>
<td>LPTIM Input 2 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_ETR</td>
<td>Digital input</td>
<td>LPTIM external trigger GPIO pin</td>
</tr>
<tr>
<td>LPTIM_CH1</td>
<td>Digital input/output</td>
<td>LPTIM channel 1 Input/Output GPIO pin</td>
</tr>
<tr>
<td>LPTIM_CH2</td>
<td>Digital input/output</td>
<td>LPTIM channel 2 Input/Output GPIO pin</td>
</tr>
</tbody>
</table>

Table 478. LPTIM4/5 input/output pins

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_IN1</td>
<td>Digital input</td>
<td>LPTIM Input 1 from GPIO pin on mux input 0</td>
</tr>
<tr>
<td>LPTIM_ETR</td>
<td>Digital input</td>
<td>LPTIM external trigger GPIO pin</td>
</tr>
<tr>
<td>LPTIM_OUT</td>
<td>Digital output</td>
<td>LPTIM Output GPIO pin</td>
</tr>
</tbody>
</table>

1. Some I/Os may not be available, refer to Section 46.4.2: LPTIM pins and internal signals.
### Table 479. LPTIM1/2/3 internal signals

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_pclk</td>
<td>Digital input</td>
<td>LPTIM APB clock domain</td>
</tr>
<tr>
<td>lptim_ker_ck</td>
<td>Digital input</td>
<td>LPTIM kernel clock</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_in2_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in2_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 2 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ic1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_ic1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_ic1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ic2_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 2 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_ic2_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 2 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_ic2_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input capture 2 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ext_trigx</td>
<td>Digital input</td>
<td>LPTIM external trigger input x</td>
</tr>
<tr>
<td>lptim_it</td>
<td>Digital output</td>
<td>LPTIM global interrupt</td>
</tr>
<tr>
<td>lptim_wakeup</td>
<td>Digital output</td>
<td>LPTIM wake-up event</td>
</tr>
<tr>
<td>lptim_ic1_dma</td>
<td>Digital output</td>
<td>LPTIM input capture 1 DMA request</td>
</tr>
<tr>
<td>lptim_ic2_dma</td>
<td>Digital output</td>
<td>LPTIM input capture 2 DMA request</td>
</tr>
<tr>
<td>lptim_ue_dma</td>
<td>Digital output</td>
<td>LPTIM update event DMA request</td>
</tr>
</tbody>
</table>

### Table 480. LPTIM4/5 internal signals

<table>
<thead>
<tr>
<th>Names</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_pclk</td>
<td>Digital input</td>
<td>LPTIM APB clock domain</td>
</tr>
<tr>
<td>lptim_ker_ck</td>
<td>Digital input</td>
<td>LPTIM kernel clock</td>
</tr>
<tr>
<td>lptim_in1_mux1</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 1</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 2</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Digital input</td>
<td>Internal LPTIM input 1 connected to mux input 3</td>
</tr>
<tr>
<td>lptim_ext_trigx</td>
<td>Digital input</td>
<td>LPTIM external trigger input x</td>
</tr>
<tr>
<td>lptim_out</td>
<td>Digital output</td>
<td>LPTIM counter output</td>
</tr>
<tr>
<td>lptim_it</td>
<td>Digital output</td>
<td>LPTIM global interrupt</td>
</tr>
<tr>
<td>lptim_wakeup</td>
<td>Digital output</td>
<td>LPTIM wake-up event</td>
</tr>
</tbody>
</table>
### 46.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

#### Table 481. LPTIM1/2/3/4/5 external trigger connection

<table>
<thead>
<tr>
<th>TRIGSEL</th>
<th>External trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ext_trig0</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>lptim_ext_trig1</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>lptim_ext_trig2</td>
<td>lptim4_out</td>
</tr>
<tr>
<td>lptim_ext_trig3</td>
<td>lptim5_out</td>
</tr>
<tr>
<td>lptim_ext_trig4</td>
<td>lptim2_ch1</td>
</tr>
<tr>
<td>lptim_ext_trig5</td>
<td>lptim3_ch1</td>
</tr>
<tr>
<td>lptim_ext_trig6</td>
<td>lptim4_out</td>
</tr>
<tr>
<td>lptim_ext_trig7</td>
<td>lptim5_out</td>
</tr>
</tbody>
</table>

#### Table 482. LPTIM1/2/3 input 1 connection

<table>
<thead>
<tr>
<th>lptim_in1_mux0</th>
<th>lptim1/2/3 input 1 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in1_mux1</td>
<td>GPIIO</td>
</tr>
<tr>
<td>lptim_in1_mux2</td>
<td>Reserved</td>
</tr>
<tr>
<td>lptim_in1_mux3</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Table 483. LPTIM1/2/3 input 2 connection

<table>
<thead>
<tr>
<th>lptim_in2_mux0</th>
<th>lptim1/2 input 2 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_in2_mux1</td>
<td>GPIIO</td>
</tr>
<tr>
<td>lptim_in2_mux2</td>
<td>Reserved</td>
</tr>
<tr>
<td>lptim_in2_mux3</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Table 484. LPTIM1/2/3 input capture 1 connection

<table>
<thead>
<tr>
<th>lptim_ic1_mux0</th>
<th>lptim1/2 input capture 1 connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>lptim_ic1_mux1</td>
<td>GPIIO</td>
</tr>
<tr>
<td>lptim_ic1_mux2</td>
<td>Reserved</td>
</tr>
<tr>
<td>lptim_ic1_mux3</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
46.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM may run in one of these two possible configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or Pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.
46.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits.
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. Figure 656 shows an example of glitch filter behavior in case of a 2 consecutive samples programmed.

![Figure 656. Glitch filter timing diagram](image)

Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to '0'. In that case, an external analog filter may be used to protect the LPTIM external inputs against glitches.

46.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] 3-bit field. The table below lists all the possible division ratios:

<table>
<thead>
<tr>
<th>programming</th>
<th>dividing factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>/1</td>
</tr>
<tr>
<td>001</td>
<td>/2</td>
</tr>
</tbody>
</table>
46.4.7 Trigger multiplexer

The LPTIM counter may be started either by software or after the detection of an active edge on one of the 8 trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:
- When TRIGEN[1:0] equals ‘00’, The LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than ‘00’, TRIGSEL[2:0] is used to select which of the 8 trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. So after a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

46.4.8 Operating mode

The LPTIM features two operating modes:
- The Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.
In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in Figure 657.

**Figure 657. LPTIM output waveform, single counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)**

<table>
<thead>
<tr>
<th>LPTIM_RCR</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repetition counter</td>
<td>2</td>
</tr>
<tr>
<td>LPTIM_ARR</td>
<td></td>
</tr>
<tr>
<td>Compare</td>
<td></td>
</tr>
<tr>
<td>PWM</td>
<td></td>
</tr>
</tbody>
</table>

- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFRG register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in Figure 658.

**Figure 658. LPTIM output waveform, Single counting mode configuration and Set-once mode activated (WAVE bit is set)**

In case of software start (TRIGEN[1:0] = '00'), the SNGSTRT setting starts the counter for one-shot counting.

**Continuous mode**

To enable the continuous counting, the CNTSTRT bit must be set.
In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in Figure 659.

In case of software start (TRIGEN[1:0] = '00'), setting CNTSTRT starts the counter for continuous counting.

Figure 659. LPTIM output waveform, Continuous counting mode configuration

SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (The ENABLE bit is set to ‘1’). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

46.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

46.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output.

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a
match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see Section 46.4.19: PWM mode.

- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require that the LPTIM_ARR register value be strictly greater than the LPTIM_CCRx register value.

The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to '0' forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to '1' forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is reconfigured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by 2 can be generated. Figure 660 below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

**Figure 660. Waveform generation**

![Waveform generation diagram](MS32467V2)
46.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to ‘0’, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to ‘1’, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

46.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM Input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- **CKSEL = 0**: the LPTIM is clocked by an internal clock source
  - **COUNTMODE = 0**
    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - **COUNTMODE = 1**
    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.
    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock
provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRES[2:0] = 000).

- **CKSEL = 1**: the LPTIM is clocked by an external clock source

  COUNTMODE value is don't care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external Input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external Input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

### 46.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

### 46.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register to zero, two reset mechanisms are implemented:

- **The synchronous reset mechanism**: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is trigger and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

- **The asynchronous reset mechanism**: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.

  Note that to read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

---

**Warning:** There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. So
46.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two Down and Up flags in the LPTIM_ISR register. Also, an interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to ‘1’. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder’s position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

<table>
<thead>
<tr>
<th>Active edge</th>
<th>Level on opposite signal (Input1 for Input2, Input2 for Input1)</th>
<th>Input1 signal</th>
<th>Input2 signal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rising</td>
<td>Falling</td>
<td>Rising</td>
</tr>
<tr>
<td>Rising Edge</td>
<td>High</td>
<td>Down</td>
<td>No count</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>No count</td>
</tr>
<tr>
<td>Falling Edge</td>
<td>High</td>
<td>No count</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>No count</td>
<td>Down</td>
</tr>
<tr>
<td>Both Edges</td>
<td>High</td>
<td>Down</td>
<td>Up</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td>Up</td>
<td>Down</td>
</tr>
</tbody>
</table>

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.
Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to ‘0’. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be ‘000’).

Figure 661. Encoder mode counting sequence

46.4.16 Repetition Counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in Figure 662.
A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to section "Register update" for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, that new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain so that the new written value is loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

**Note:** After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the REPOK flag is set, leads to unpredictable results.

**Caution:** When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.
46.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

46.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the IC DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value...
(PRES2:0 field in CFGR register), there is a variable latency that leads to a systematic offset (see Table 490) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRES field in CFGR register)

**Example**: determining the real capture value when PRES2:0 = 0x2 and ICxF = 0x3.

For this configuration (PRES2:0 = 0x2 and ICxF = 0x3) and according to the Table 490, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

<table>
<thead>
<tr>
<th>Prescaler PRES2:0</th>
<th>ICxF[1:0]</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
### 46.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

*Figure 665* gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

<table>
<thead>
<tr>
<th>Prescaler PRES[2:0]</th>
<th>ICxF[1:0]</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.
Figure 666 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 666. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)

PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. Figure 667 illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.
46.4.20 DMA requests

The LPTIM has the capability to generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests are used to re-program part of the LPTIMER, multiple times, at regular intervals, without software overhead.

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the CCRx register. The captured values in CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:
- if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)
- or if the channel x is disabled (clear CCxE bit)
- or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

Note: For both PWM modes, the compare match, auto-reload match and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 665 illustrates this behavior.
The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

**Note:** The UE DMA request signal lptim_ue_dma is reset in the following conditions:
- if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)
- or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)
- or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled

### 46.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

### 46.5 LPTIM low-power modes

**Note:** All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

### 46.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

**Note:** If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (Status Register) is set, the interrupt is not asserted.
Table 491. Interrupt events

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop mode(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIMx</td>
<td>Compare match</td>
<td>CCxIF</td>
<td>CCxIE</td>
<td>Write 1 to CCxCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Input capture(2)</td>
<td>CCxIF</td>
<td>CCxIE</td>
<td>Write 1 to CCxCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Over-capture(2)</td>
<td>CCxOF</td>
<td>CCxOIE</td>
<td>Write 1 to CCxOCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Auto-reload match</td>
<td>ARRM</td>
<td>ARRMIE</td>
<td>Write 1 to ARRMCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>External trigger event</td>
<td>EXTTRIG</td>
<td>EXTTRIGIE</td>
<td>Write 1 to EXTTRIGCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Auto-reload register update OK</td>
<td>ARROK</td>
<td>ARROKIE</td>
<td>Write 1 to ARROKCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Capture/compare register update OK</td>
<td>CMPxOK</td>
<td>CMPxOKIE</td>
<td>Write 1 to CMPxOKCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Direction change to up(3)</td>
<td>UP</td>
<td>UPIE</td>
<td>Write 1 to UPCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Direction change to down(3)</td>
<td>DOWN</td>
<td>DOWNIE</td>
<td>Write 1 to DOWNCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Update Event</td>
<td>UE</td>
<td>UEIE</td>
<td>Write 1 to UECF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Repetition register update OK</td>
<td>REPOK</td>
<td>REPOKIE</td>
<td>Write 1 to REPOKCF</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to Section 46.3: LPTIM implementation.
2. If LPTIM does not implement any channel this event does not exist. Refer to Section 46.3: LPTIM implementation.
3. If LPTIM does not support encoder mode feature, this event does not exist. Refer to Section 46.3: LPTIM implementation.

46.7 LPTIM registers

Refer to Section 1.2: List of abbreviations for registers on page 120 for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).
46.7.1 LPTIMx interrupt and status register (LPTIMx_ISR)(x = 4 to 5)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK**: Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3: LPTIM implementation.*

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3: LPTIM implementation.*

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.
Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

- 0: No match
- 1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

### 46.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR)

(x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td><strong>DIEROK</strong>: Interrupt enable register update OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.</td>
<td></td>
</tr>
<tr>
<td>23-21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td><strong>CMP2OK</strong>: Compare register 2 update OK</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong>: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.</td>
<td></td>
</tr>
<tr>
<td>18-12</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>11-8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>7-4</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>3-0</td>
<td><strong>CC2IF</strong>, <strong>REP OK</strong>, <strong>UE</strong>, <strong>DOWN</strong>, <strong>UP</strong>, <strong>ARR OK</strong>, <strong>CMP1 OK</strong>, <strong>EXT TRIG</strong>, <strong>ARRM</strong>, <strong>CC1IF</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>These flags are set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The flags can be cleared by writing 1 to the corresponding bits in the LPTIM_ICR register.</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

- **Note**: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.
Bit 11  Reserved, must be kept at reset value.

Bit 10  Reserved, must be kept at reset value.

Bit 9  **CC2IF:** Compare 2 interrupt flag

  **If channel CC2 is configured as output:**
  The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.
  0: No match
  1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

  **Note:** If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.

Bit 8  **REPOK:** Repetition register update OK

  REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7  **UE:** LPTIM update event occurred

  UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6  **DOWN:** Counter direction change up to down

  In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

  **Note:** If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 5  **UP:** Counter direction change down to up

  In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

  **Note:** If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 4  **ARROK:** Autoreload register update OK

  ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3  **CMP1OK:** Compare register 1 update OK

  CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.
Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

**If channel CC1 is configured as output:**
The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match
1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

### 46.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>4</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>

**Bits 31:25**: Reserved, must be kept at reset value.

**Bit 24** **DIEROK**: Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

**Bits 23:16**: Reserved, must be kept at reset value.

**Bit 15**: Reserved, must be kept at reset value.

**Bit 14**: Reserved, must be kept at reset value.

**Bit 13** **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.
1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

**Note**: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.
Bit 12 **CC1OF**: Capture 1 over-capture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.
0: No over-capture has been detected.
1: The counter value has been captured in **LPTIM_CCR1** register while CC1IF flag was already set.
*Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag
If channel CC2 is configured as input:
CC2IF is set by hardware to inform application that the current value of the counter is captured in **LPTIM_CCR2** register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.
0: No input capture occurred
1: The counter value has been captured in the **LPTIM_CCR2** register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 8 **REPOK**: Repetition register update OK
REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred
UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM蒋CR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down
In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 5 **UP**: Counter direction change down to up
In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 4 **ARROK**: Autoreload register update OK
ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.
Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register’s value reached the LPTIM_ARR register’s value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred
1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

### 46.7.4 LPTIMx interrupt clear register (LPTIMx_ICR)(x = 4 to 5)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF**: Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clears the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clears the DOWN flag in the LPTIM_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clears the UP flag in the LPTIM_ISR register.

*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*
**46.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR)**

(x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

| Bit 31:25 Reserved, must be kept at reset value. |
| Bit 24 **DIEROKCF**: Interrupt enable register update OK clear flag  
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register. |
| Bit 23:22 Reserved, must be kept at reset value. |
| Bit 21 Reserved, must be kept at reset value. |
| Bit 20 Reserved, must be kept at reset value. |
| Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag  
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.  
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.* |
| Bit 18:12 Reserved, must be kept at reset value. |
| Bit 11 Reserved, must be kept at reset value. |
| Bit 10 Reserved, must be kept at reset value. |
| Bit 9 **CC2CF**: Capture/compare 2 clear flag  
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.  
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.* |
Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register.

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register.

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register.

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

### 46.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>CC2 OCF</td>
<td>CC1 OCF</td>
<td>Res</td>
<td>Res</td>
<td>CC2CF</td>
<td>REPOK CF</td>
<td>UECF</td>
<td>DOWN CF</td>
<td>UPDF</td>
<td>ARRO KCF</td>
<td>EXTTR IGCF</td>
<td>ARRM CF</td>
<td>CC1CF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF**: Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.
Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag  
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.  
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag  
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.  
*Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag  
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.  
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 8 **REPOKCF**: Repetition register update OK clear flag  
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag  
Writing 1 to this bit clears the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag  
Writing 1 to this bit clears the DOWN flag in the LPTIM_ISR register.  
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 5 **UPCF**: Direction change to UP clear flag  
Writing 1 to this bit clears the UP flag in the LPTIM_ISR register.  
*Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.*

Bit 4 **ARROKCF**: Autoreload register update OK clear flag  
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag  
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register.

Bit 1 **ARRMCF**: Autoreload match clear flag  
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register.

Bit 0 **CC1CF**: Capture/compare 1 clear flag  
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

### 46.7.7 LPTIMx interrupt enable register (LPTIMx_DIER)(x = 4 to 5)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

2102/3791  
RM0477 Rev 6
Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.
### 46.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable

0: UE DMA request disabled. Writing ‘0’ to the UEDE bit resets the associated ue_dma_req signal.

1: UE DMA request enabled

*Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.*

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable

0: CMPOK register 2 interrupt disabled

1: CMPOK register 2 interrupt enabled

*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable

0: Capture/compare 2 interrupt disabled

1: Capture/compare 2 interrupt enabled

*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable

0: Repetition register update OK interrupt disabled

1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable

0: Update event interrupt disabled

1: Update event interrupt enabled
**Caution:** The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

### 46.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 6</th>
<th>DOWNIE</th>
<th>Direction change to down Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DOWN interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>DOWN interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.

<table>
<thead>
<tr>
<th>Bit 5</th>
<th>UPIE</th>
<th>Direction change to UP Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UP interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>UP interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.

<table>
<thead>
<tr>
<th>Bit 4</th>
<th>ARROKIE</th>
<th>Autoreload register update OK Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ARROK interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ARROK interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>CMP10KIE</th>
<th>Compare register 1 update OK interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>CMPOK register 1 interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>CMPOK register 1 interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 2</th>
<th>EXTRIGIE</th>
<th>External trigger valid edge Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>EXTRIG interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>EXTRIG interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 1</th>
<th>ARRMIE</th>
<th>Autoreload match Interrupt Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ARRM interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ARRM interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>CC1IE</th>
<th>Capture/compare 1 interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Capture/compare 1 interrupt disabled</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Capture/compare 1 interrupt enabled</td>
<td></td>
</tr>
</tbody>
</table>

### Description of the register

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:28  Reserved, must be kept at reset value.

Bit 27  Reserved, must be kept at reset value.
Bit 26  Reserved, must be kept at reset value.

Bit 25  **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 24  Reserved, must be kept at reset value.

Bit 23  **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
*Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.*

Bits 22:17  Reserved, must be kept at reset value.

Bit 16  **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
*Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.*

Bit 15  Reserved, must be kept at reset value.

Bit 14  Reserved, must be kept at reset value.

Bit 13  **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 12  **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
*Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 46.3.*

Bit 11  Reserved, must be kept at reset value.

Bit 10  Reserved, must be kept at reset value.

Bit 9  **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
*Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3.*

Bit 8  **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7  **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled
Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

46.7.10 LPTIM configuration register (LPTIM_CFRG)
Address offset: 0x00C
Reset value: 0x0000 0000

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

46.7.10 LPTIM configuration register (LPTIM_CFRG)
Address offset: 0x00C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Reset Value</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 31:30</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bits 28:25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 24 ENC: Encoder mode enable
   The ENC bit controls the Encoder mode
   0: Encoder mode disabled
   1: Encoder mode enabled

Note: If the LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3.

Bit 23 COUNTMODE: counter mode enabled
   The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
   0: the counter is incremented following each internal clock pulse
   1: the counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 PRELOAD: Registers update mode
   The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
   0: Registers are updated after each APB bus write access
   1: Registers are updated at the end of the current LPTIM period

Bit 21 WAVPOL: Waveform shape polarity
   The WAVPOL bit controls the output polarity
   0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers
   1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers

Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 46.3.

Bit 20 WAVE: Waveform shape
   The WAVE bit controls the output shape
   0: Deactivate Set-once mode
   1: Activate the Set-once mode

Bit 19 TIMOUT: Timeout enable
   The TIMOUT bit controls the Timeout feature
   0: A trigger event arriving when the timer is already started is ignored
   1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 TRIGEN[1:0]: Trigger enable and polarity
   The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
   00: software trigger (counting start is initiated by software)
   01: rising edge is the active edge
   10: falling edge is the active edge
   11: both edges are active edges

Bit 16 Reserved, must be kept at reset value.
Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

- 000: lptim_ext_trig0
- 001: lptim_ext_trig1
- 010: lptim_ext_trig2
- 011: lptim_ext_trig3
- 100: lptim_ext_trig4
- 101: lptim_ext_trig5
- 110: lptim_ext_trig6
- 111: lptim_ext_trig7

See Section 46.4.3: LPTIM input and trigger mapping for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]**: Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

- 000: /1
- 001: /2
- 010: /4
- 011: /8
- 100: /16
- 101: /32
- 110: /64
- 111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]**: Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature.

- 00: any trigger active level change is considered as a valid trigger
- 01: trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
- 10: trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
- 11: trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]**: Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

- 00: any external clock signal level change is considered as a valid transition
- 01: external clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
- 10: external clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
- 11: external clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.
Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to '0').

### 46.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010  
Reset value: 0x0000 0000

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

**Bit 4** RSTARE: Reset after read enable  
This bit is set and cleared by software. When RSTARE is set to ‘1’, any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.  
This bit can be set only when the LPTIM is enabled.

**Bit 3** COUNTRST: Counter reset  
This bit is set by software and cleared by hardware. When set to ‘1’ this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock may be different from APB clock).  
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to ‘1’ by software before it is already cleared to ‘0’ by hardware. Software must consequently check that COUNTRST bit is already cleared to ‘0’ before attempting to set it to ‘1’.
Bit 2  **CNTSTRT**: Timer start in Continuous mode
This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in Continuous mode.
If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.
If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1  **SNGSTRT**: LPTIM start in Single mode
This bit is set by software and cleared by hardware.
In case of software start (TRIGEN[1:0] = '00'), setting this bit starts the LPTIM in single pulse mode.
If the software start is disabled (TRIGEN[1:0] different than '00'), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.
If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.
This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0  **ENABLE**: LPTIM enable
The ENABLE bit is set and cleared by software.
0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).
1: LPTIM is enabled

### 46.7.12  LPTIM compare register 1 (LPTIM_CCR1)
Address offset: 0x014
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **CCR1[15:0]**: Capture/compare 1 value

**If channel CC1 is configured as output:**
CCR1 is the value to be loaded in the capture/compare 1 register.
Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PRELOAD bit is reset.
The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

**If channel CC1 is configured as input:**
CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

**If LPTIM does not implement any channel:**
The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.
Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

46.7.13 LPTIM autoreload register (LPTIM_ARR)
Address offset: 0x018
Reset value: 0x0000 0001

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

46.7.14 LPTIM counter register (LPTIM_CNT)
Address offset: 0x01C
Reset value: 0x0000 0000

Caution: The LPTIM_CNT register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).
46.7.15 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024
Reset value: 0x0000 0000

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

Bits 21:20 **IC2SEL[1:0]:** LPTIM input capture 2 selection
The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to Section 46.4.3: LPTIM input and trigger mapping.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]:** LPTIM input capture 1 selection
The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to Section 46.4.3: LPTIM input and trigger mapping.

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

Bits 5:4 **IN2SEL[1:0]:** LPTIM input 2 selection
The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to Section 46.4.3: LPTIM input and trigger mapping.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]:** LPTIM input 1 selection
The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to Section 46.4.3: LPTIM input and trigger mapping.
46.7.16  LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  REP[7:0]: Repetition register value

REP is the repetition value for the LPTIM.

Caution:  The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

46.7.17  LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C
Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  REP[7:0]: Repetition register value

REP is the repetition value for the LPTIM.
Bits 31:30  Reserved, must be kept at reset value.

Bits 29:28  **IC2F[1:0]: Input capture 2 filter**
This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.
- 00: any external input capture signal level change is considered as a valid transition
- 01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
- 10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
- 11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:24  **IC2PSC[1:0]: Input capture 2 prescaler**
This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).
- 00: no prescaler, capture is done each time an edge is detected on the capture input
- 01: capture is done once every 2 events
- 10: capture is done once every 4 events
- 11: capture is done once every 8 events

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

Bits 19:18  **CC2P[1:0]: Capture/compare 2 output polarity.**
**Condition: CC2 as output**
Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.
- 0: OC2 active high
- 1: OC2 active low
**Condition: CC2 as input**
This field is used to select the IC2 polarity for capture operations.
- 00: rising edge, circuit is sensitive to IC2 rising edge
- 01: falling edge, circuit is sensitive to IC2 falling edge
- 10: reserved, do not use this configuration.
- 11: both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17  **CC2E: Capture/compare 2 output enable.**
**Condition: CC2 as output**
- 0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.
- 1: On - OC2 signal is output on the corresponding output pin
**Condition: CC2 as input**
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.
- 0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.
- 1: Capture enabled.

Bit 16  **CC2SEL: Capture/compare 2 selection**
This bitfield defines the direction of the channel, input (capture) or output mode.
- 0: CC2 channel is configured in output PWM mode
- 1: CC2 channel is configured in input capture mode

Bits 15:14  Reserved, must be kept at reset value.
Low-power timer (LPTIM)  RM0477

Bits 13:12  **IC1F[1:0]**: Input capture 1 filter
This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.
00: any external input capture signal level change is considered as a valid transition
01: external input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: external input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: external input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:8  **IC1PSC[1:0]**: Input capture 1 prescaler
This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:2  **CC1P[1:0]**: Capture/compare 1 output polarity.
**Condition: CC1 as output**
Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.
0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers
1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

**Condition: CC1 as input**
This field is used to select the IC1 polarity for capture operations.
00: rising edge, circuit is sensitive to IC1 rising edge
01: falling edge, circuit is sensitive to IC1 falling edge
10: reserved, do not use this configuration.
11: both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1  **CC1E**: Capture/compare 1 output enable.
**Condition: CC1 as output**
0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.
1: On - OC1 signal is output on the corresponding output pin

**Condition: CC1 as input**
This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.
0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.
1: Capture enabled.

Bit 0  **CC1SEL**: Capture/compare 1 selection
This bitfield defines the direction of the channel input (capture) or output mode.
0: CC1 channel is configured in output PWM mode
1: CC1 channel is configured in input capture mode

**Caution:** After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)
kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,...128). Any successive write violating this delay, leads to unpredictable results.

**Caution:** The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

*If LPTIM does not implement any channel this register is reserved. Refer to Section 46.3.*

**46.7.18 LPTIM compare register 2 (LPTIM_CCR2)**

Address offset: 0x034

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**CCR2[15:0]**

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]:** Capture/compare 2 value

*If channel CC2 is configured as output:*

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

*If channel CC2 is configured as input:*

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event.

The LPTIM_CCR2 register is read-only and cannot be programmed.

**Caution:** The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to ‘1’).

**Note:** If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 46.3: LPTIM implementation.*

**46.7.19 LPTIM register map**

The following table summarizes the LPTIM registers.

*Table 492. LPTIM register map and reset values*

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>LPTIMx_ISR (x = 4 to 5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 492. LPTIM register map and reset values (continued)

| Offset  | Register name | `x` | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000   | LPTIMx_ISR    |
|         | (x = 1 to 3)  |
|         | Output compare mode |
|         | Reset value     |
|         | 0x000           | DIPR | 0000000000000000 |
| 0x004   | LPTIMx_ISR    |
|         | (x = 4 to 5)  |
|         | Input capture mode |
|         | Reset value     |
|         | 0x000           | DIPR | 0000000000000000 |
| 0x008   | LPTIMx_DIER   |
|         | (x = 1 to 3)  |
|         | Output compare mode |
|         | Reset value     |
|         | 0x000           | DIPR | 0000000000000000 |
| 0x00C   | LPTIM_CFRG    |
|         |                  |
|         | Reset value     |
|         | 0x000           | ENC | 0000000000000000 |
| 0x010   | LPTIM_CR       |
|         |                  |
|         | Reset value     |
|         | 0x000           | CCR1 | 0000000000000000 |
### Table 492. LPTIM register map and reset values (continued)

| Offset | Register name            | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|--------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x018  | LPTIM_ARR                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 1   |
|        | Reset value              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x01C  | LPTIM_CNT                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 1   |
|        | Reset value              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x024  | LPTIM_CFR2               |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     | 0   |
|        | Reset value              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x028  | LPTIM_RCR                |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x02C  | LPTIM_CCMR1\(^{1(4)}\)  | IC2F[1:0]|     | IC2SEL[1:0]| IC1F[1:0]| IC1SEL[1:0]| IC1PSC[1:0]| CC1P[1:0]| CC1E| CC1SEL| IC2PSC[1:0]| IC2SEL[1:0]| IC2F[1:0]|     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value              | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |     |     |     |     |     |
| 0x034  | LPTIM_CCR2\(^{(5)}\)     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value              |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 46.3: LPTIM implementation.
2. If LPTIM does not support encoder mode feature, this bit is reserved. Refer to Section 46.3: LPTIM implementation.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 46.3: LPTIM implementation.
4. If LPTIM does not implement any channel this register is reserved. Refer to Section 46.3: LPTIM implementation.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to Section 46.3: LPTIM implementation.

Refer to Section 2.3 on page 149 for the register boundary addresses.
47 System window watchdog (WWDG)

47.1 Introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time-window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

47.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see Figure 669)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

47.3 WWDG implementation

<table>
<thead>
<tr>
<th>WWDG mode / feature</th>
<th>WWDG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window function</td>
<td>X</td>
</tr>
<tr>
<td>Early wake-up interrupt generation</td>
<td>X</td>
</tr>
<tr>
<td>System reset generation</td>
<td>X</td>
</tr>
<tr>
<td>Capability to work in system Stop</td>
<td>-</td>
</tr>
<tr>
<td>Capability to work in system Standby</td>
<td>-</td>
</tr>
<tr>
<td>Capability to be frozen when the microcontroller enters in Debug mode</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the Hardware mode</td>
<td>X</td>
</tr>
</tbody>
</table>

1. "X" = supported, "-" = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_WWDG_STOP in DBG block.
47.4 **WWDG functional description**

If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to **Figure 668** and to **Section 47.4.2: WWDG internal signals** for the WWDG block diagram.

**47.4.1 WWDG block diagram**

![Watchdog block diagram](image)

**47.4.2 WWDG internal signals**

*Table 494* gives the list of WWDG internal signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pclk</td>
<td>Digital input</td>
<td>APB bus clock</td>
</tr>
<tr>
<td>wwdg_out_rst</td>
<td>Digital output</td>
<td>WWDG reset signal output</td>
</tr>
<tr>
<td>wwdg_it</td>
<td>Digital output</td>
<td>WWDG early interrupt output</td>
</tr>
</tbody>
</table>
47.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

47.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see Figure 669). The **WWDG configuration register (WWDG_CFR)** contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. **Figure 669** describes the window watchdog process.

*Note:* The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

47.4.5 How to program the watchdog timeout

Use the formula in **Figure 669** to calculate the WWDG timeout.

---

**Warning:** When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.
The formula to calculate the timeout value is given by:

\[ t_{WWDG} = t_{PCLK} \times \frac{4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1)}{4096 \times 2^3} \] (ms)

where:
- \( t_{WWDG} \): WWDG timeout
- \( t_{PCLK} \): APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

\[ t_{WWDG} = \frac{1}{48000} \times 4096 \times 2^3 \times (63 + 1) = 43.69\text{ms} \]

Refer to the datasheet for the minimum and maximum values of \( t_{WWDG} \).

### 47.4.6 Debug mode

When the CPU enters debug mode, WWDG counter either continues to work normally or stops, depending on debug settings. For more details refer to Section 47.3: WWDG implementation and to Section 66: Debug infrastructure.
47.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing '0' to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

47.6 WWDG registers

Refer to Section 1.2 on page 120 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

47.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000
Reset value: 0x0000 007F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bit 7 WDGA: Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled
1: Watchdog enabled
47.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004
Reset value: 0x0000 007F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **WDGTB[2:0]**: Timer base
  - 000: CK counter clock (PCLK div 4096) div 1
  - 001: CK counter clock (PCLK div 4096) div 2
  - 010: CK counter clock (PCLK div 4096) div 4
  - 011: CK counter clock (PCLK div 4096) div 8
  - 100: CK counter clock (PCLK div 4096) div 16
  - 101: CK counter clock (PCLK div 4096) div 32
  - 110: CK counter clock (PCLK div 4096) div 64
  - 111: CK counter clock (PCLK div 4096) div 128

- **Bit 10**: Reserved, must be kept at reset value.

- **Bit 9** **EWI**: Early wake-up interrupt enable
  - Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

- **Bits 8:7**: Reserved, must be kept at reset value.

- **Bits 6:0** **W[6:0]**: 7-bit window value
  - These bits contain the window value to be compared with the down-counter.

47.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **WDGTB[2:0]**: Timer base (continued)
- **Bit 10**: Reserved, must be kept at reset value.
- **Bit 9** **EWI**: Early wake-up interrupt enable (continued)
- **Bits 8:7**: Reserved, must be kept at reset value.
- **Bits 6:0** **W[6:0]**: 7-bit window value (continued)
  - These bits contain the window value to be compared with the down-counter.
Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF**: Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

### 47.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

**Table 495. WWDG register map and reset values**

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | WWDG_CR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x004  | WWDG_CFR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | WWDG_SR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to *Section 2.3 on page 149* for the register boundary addresses.
48 Independent watchdog (IWDG)

48.1 Introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures. The IWDG is clocked by an independent clock, and stays active even if the main clock fails. In addition, the watchdog function is performed in the VDD voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to Section 48.3 to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

48.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

48.3 IWDG implementation

Table 496. IWDG features (1)

<table>
<thead>
<tr>
<th>IWDG modes/features</th>
<th>IWDG</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSI used as IWDG kernel clock (iwdg_ker_ck)</td>
<td>X</td>
</tr>
<tr>
<td>Window function</td>
<td>X</td>
</tr>
<tr>
<td>Early wake-up interrupt generation</td>
<td>X</td>
</tr>
<tr>
<td>System reset generation (2)</td>
<td>X</td>
</tr>
<tr>
<td>Capability to work in system Stop</td>
<td>X</td>
</tr>
<tr>
<td>Capability to work in system Standby</td>
<td>X</td>
</tr>
<tr>
<td>Capability to generate an interrupt in system Stop</td>
<td>X</td>
</tr>
<tr>
<td>Capability to generate an interrupt in system Standby</td>
<td>-</td>
</tr>
<tr>
<td>Capability to be frozen when the microcontroller enters in Debug mode (3)</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the activity in Stop mode (4)</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the activity in Standby mode (5)</td>
<td>X</td>
</tr>
<tr>
<td>Option bytes to control the Hardware mode (6)</td>
<td>X</td>
</tr>
</tbody>
</table>

1. ‘X’ = supported, ‘-’ = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

48.4  IWDG functional description

48.4.1  IWDG block diagram

*Figure 670* shows the functional blocks of the independent watchdog module.

*Figure 670. Independent watchdog block diagram*

The register and IRQ interfaces are located into the V\textsubscript{CORE} voltage domain. The watchdog function itself is located into the V\textsubscript{DD} voltage domain to remain functional in low power modes. See *Section 48.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V\textsubscript{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.
48.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in Table 497.

Table 497. IWDG internal input/output signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>iwdg_ker_ck</td>
<td>Input</td>
<td>IWDG kernel clock</td>
</tr>
<tr>
<td>iwdg_ker_req</td>
<td>Input</td>
<td>IWDG kernel clock request</td>
</tr>
<tr>
<td>iwdg_pclk</td>
<td>Input</td>
<td>IWDG APB clock</td>
</tr>
<tr>
<td>iwdg_out_rst</td>
<td>Output</td>
<td>IWDG reset output</td>
</tr>
<tr>
<td>iwdg_in_rst</td>
<td>Input</td>
<td>IWDG reset input</td>
</tr>
<tr>
<td>iwdg_wkup</td>
<td>Output</td>
<td>IWDG wake-up event</td>
</tr>
<tr>
<td>iwdg_it</td>
<td>Output</td>
<td>IWDG early wake-up interrupt</td>
</tr>
</tbody>
</table>

48.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The Software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the IWDG key register (IWDG_KR), and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see Section 48.3 for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

Whenever the key value 0x0000 AAAA is written in the IWDG key register (IWDG_KR), the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in Figure 671, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].
Figure 671. Reset timing due to timeout

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

48.4.4 Window option

The IWDG can also work as a window watchdog, by setting the appropriate window in the IWDG window register (IWDG_WINR).

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the IWDG window register (IWDG_WINR). As shown in Figure 672, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the IWDG window register (IWDG_WINR) is 0x0000 0xFFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.
Figure 672. Reset timing due to refresh in the not allowed area

Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG key register (IWDG_KR).
2. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
3. Write the IWDG prescaler by programming IWDG prescaler register (IWDG_PR).
4. Write the IWDG reload register (IWDG_RLR).
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the IWDG early wake-up interrupt register (IWDG_EWCR).
6. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into IWDG key register (IWDG_KR) to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the IWDG key register (IWDG_KR).
2. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
3. Write the prescaler by programming the IWDG prescaler register (IWDG_PR).
4. Write the IWDG reload register (IWDG_RLR).
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the IWDG early wake-up interrupt register (IWDG_EWCR).
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into IWDG key register (IWDG_KR).

Updating the window comparator

It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 673 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high. The new window value and the reload of IWDCNT with RL[11:0] are effective on the next rising edge of presc_ck. The WVU flag goes back to 0, in the worst case, two kernel clock periods later. So WVU remains high at most one period of presc_ck, plus two periods of the kernel clock.
48.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to Section 48.3 for details on the capabilities of this product.

48.4.6 Register access protection

Write accesses to **IWDG prescaler register (IWDG_PR)**, **IWDG reload register (IWDG_RLR)**, **IWDG early wake-up interrupt register (IWDG_EWCR)** and **IWDG window register (IWDG_WINR)** are protected. To modify them, first write 0x0000 5555 in the **IWDG key register (IWDG_KR)**. A write access to this register with a different value breaks the sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

48.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to Section 48.3 for details.
The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the IWDG early wake-up interrupt register (IWDG_EWCR) to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the IWDG status register (IWDG_SR) is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the IWDG early wake-up interrupt register (IWDG_EWCR).

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].
The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

**Changing the early wake-up comparator value**

It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the IWDG early wake-up interrupt register (IWDG_EWCR).
3. Wait for EWU = 0, EWU is located into the IWDG status register (IWDG_SR).
4. Write-protect registers by writing 0x0000 0000 to IWDG key register (IWDG_KR).

Step 3 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

*Figure 674* shows this sequence. As soon as the IWDG_EWCR register is written, the EWU flag goes high. The new comparator value and the reload of IWDCNT with RL[11:0] are effective on the next rising edge of presc_ck. The EWU flag goes back to 0, in the worst case, two kernel clock periods later. So, EWU remains high at most one period of presc_ck, plus two periods of the kernel clock.
48.7 IWDG registers

Refer to Section 1.2 on page 120 for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the VDD voltage domain. When the \texttt{iwdg\_in\_rst} is asserted, the watchdog logic and the shadow registers located into the VDD voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

---

Table 499 summarizes the IWDG interrupt request.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Interrupt method</th>
<th>Interrupt enable control bit</th>
<th>Activated interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>IWDTCNT reaches EWIT value</td>
<td>EWIF</td>
<td>Writing EWIC to 1</td>
<td>EWIE</td>
<td>(Y(1))</td>
</tr>
</tbody>
</table>

1. Generated when a clock is present on \texttt{iwdg\_pclk} input.
2. Generated when a clock is present on \texttt{iwdg\_ker\_ck} input.
48.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| KEY[15:0] |
| w w w w w w w w w w w w w w w w |

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:
- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

48.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| PR[3:0] |
| rw rw rw rw |

Bits 31:4 Reserved, must be kept at reset value.
48.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08
Reset value: 0x0000 0FFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 RL[11:0]: Watchdog counter reload value

These bits are write access protected, see Section 48.4.6. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the IWDG key register (IWDG_KR). The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the IWDG status register (IWDG_SR) must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

48.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C
Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in
software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **EWIF**: Watchdog early interrupt flag
This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_EWCR register to ‘1’.

Bits 13:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit
Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to ‘1’ until the IWDG is reset. In hardware mode, this bit is always set to ‘1’.

0: The IWDG is not activated
1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update
This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the VDD voltage domain (takes up to one period of presc_ck and two periods of the IWDG kernel clock iwdg_ker_ck).
The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU**: Watchdog counter window value update
This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the VDD voltage domain (takes up to one period of presc_ck and two periods of the IWDG kernel clock iwdg_ker_ck).
The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU**: Watchdog counter reload value update
This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the VDD voltage domain (takes up to six periods of the IWDG kernel clock iwdg_ker_ck).
The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU**: Watchdog prescaler value update
This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the VDD voltage domain (takes up to six periods of the IWDG kernel clock iwdg_ker_ck).
The prescaler value can be updated only when PVU bit is reset.

**Note:** If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the
early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

48.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]**: Watchdog counter window value

These bits are write access protected, see Section 48.4.6. They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDDCNT downcounter must be reloaded when its value is lower than **WIN[11:0] + 1** and greater than 1.

The WVU bit in the **IWDG status register (IWDG_SR)** must be reset to be able to change the reload value.

**Note**: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the **IWDG status register (IWDG_SR)** is reset.

48.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EWIE</th>
<th>EWIC</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the **IWDG status register (IWDG_SR)** must be reset to be able to change the value of this bit.
Bit 14 **EWIC**: Watchdog early interrupt acknowledge
The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has no effect, reading this flag returns a 0.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]**: Watchdog counter window value
These bits are write access protected (see Section 48.4.6). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the **IWDG status register (IWDG_SR)** must be reset to be able to change the reload value.

*Note*: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the VDD voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the **IWDG status register (IWDG_SR)** is reset.
### IWDG register map

**Table 500. IWDG register map and reset values**

| Offset | Register name | Offset name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | IWDG_KR       | KEY[15:0]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x04   | IWDG_PR       | PR[3:0]     | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x08   | IWDG_RLR      | RL[11:0]    | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0C   | IWDG_SR       | EWIF        | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x10   | IWDG_WINR     | WIN[11:0]   | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x14   | IWDG_EWCR     | EWIE        | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 149 for the register boundary addresses.
49 Real-time clock (RTC)

49.1 Introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in $V_{BAT}$ mode.

49.2 RTC main features

The RTC supports the following features (see Figure 676: RTC block diagram):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to $V_{BAT}$ mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the $V_{DD}$ supply when present or from the $V_{BAT}$ pin.

The RTC is functional in $V_{BAT}$ mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

49.3 RTC functional description

49.3.1 RTC block diagram
Figure 676. RTC block diagram

Real-time clock (RTC) RM0477
49.3.2 RTC pins and internal signals

Table 501. RTC input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Input</td>
<td>RTC timestamp input</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Input</td>
<td>RTC 50 or 60 Hz reference clock input</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Output</td>
<td>RTC output 1</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Output</td>
<td>RTC output 2</td>
</tr>
</tbody>
</table>

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- **CALIB**: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- **TAMPALRM**: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 502. RTC internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_ker_ck</td>
<td>Input</td>
<td>RTC kernel clock, also named RTCCLK in this document</td>
</tr>
<tr>
<td>rtc_pclk</td>
<td>Input</td>
<td>RTC APB clock</td>
</tr>
<tr>
<td>rtc_its</td>
<td>Input</td>
<td>RTC internal timestamp event</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>Input</td>
<td>Tamper event (internal or external) detected in TAMP peripheral</td>
</tr>
<tr>
<td>rtc_it</td>
<td>Output</td>
<td>RTC interrupts (refer to Section 49.5: RTC interrupts for details)</td>
</tr>
<tr>
<td>rtc_alra_trg</td>
<td>Output</td>
<td>RTC alarm A event detection trigger</td>
</tr>
<tr>
<td>rtc_alrb_trg</td>
<td>Output</td>
<td>RTC alarm B event detection trigger</td>
</tr>
<tr>
<td>rtc_wut_trg</td>
<td>Output</td>
<td>RTC wake-up timer event detection trigger</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>Output</td>
<td>RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.</td>
</tr>
</tbody>
</table>

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or $V_{\text{BAT}}$ when the selected clock is not LSE. Refer to Section 49.4: RTC low-power modes for more details.
The triggers outputs can be used as triggers for other peripherals.

### 49.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (\(V_{BAT}\)) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC\_OUT1, RTC\_TS, TAMP\_IN1 and TAMP\_OUT2 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in \(V_{BAT}\) mode.

The output mechanism follows the priority order shown in Table 504.

---

**Table 503. RTC interconnection**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtc_its</td>
<td>From power controller (PWR): main power loss/switch to (V_{BAT}) detection output</td>
</tr>
<tr>
<td>rtc_tamp_evt</td>
<td>From TAMP peripheral: tamp_evt</td>
</tr>
<tr>
<td>rtc_calovf</td>
<td>To TAMP peripheral: tamp_itamp5</td>
</tr>
</tbody>
</table>

---

**Table 504. RTC pin PC13 configuration**

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL[1:0] (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPALRM_PU</th>
<th>TAMP1E (TAMP_IN1 input enable)</th>
<th>T_SE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMPALRM output</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>Push-Pull</td>
<td>00</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
</tr>
<tr>
<td>PC13 Pin function</td>
<td>OSEL[1:0] (ALARM output enable)</td>
<td>TAMPOE (TAMPER output enable)</td>
<td>COE (CALIB output enable)</td>
<td>OUT2EN</td>
<td>TAMPALRM_TYPE</td>
<td>TAMPALRM_PU</td>
<td>TAMPE2AM=1 with ATOSHARE=0, or TAMPE2=x=1 and ATOSEL=x=1</td>
<td>TAMPE1 (TAMP_IN1 input enable)</td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------------------</td>
<td>------------------------------</td>
<td>--------------------------</td>
<td>--------</td>
<td>----------------</td>
<td>------------</td>
<td>------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>TAMPALRM output Open-Drain(2)</td>
<td>No pull</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>1</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>Internal pull-up</td>
<td>0</td>
<td>01 or 10 or 11</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01 or 10 or 11</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td>CALIB output PP</td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMP_OUT2 output PP</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td>TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>1</td>
<td>Don’t care</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>0</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>0</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RTC_TS and TAMP_IN1 input floating</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>00</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>01 or 10 or 11</td>
<td>0</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>0</td>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
In addition, it is possible to output RTC\_OUT2 on PB2 pin thanks to OUT2EN bit. The different functions are mapped on RTC\_OUT1 or on RTC\_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table Table 505.

### Table 504. RTC pin PC13 configuration\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>PC13 Pin function</th>
<th>OSEL([1:0]) (ALARM output enable)</th>
<th>TAMPOE (TAMPER output enable)</th>
<th>COE (CALIB output enable)</th>
<th>OUT2EN</th>
<th>TAMPALRM_TYPE</th>
<th>TAMPALRM_PU</th>
<th>TAMPE1 (TAMP_IN1 input enable)</th>
<th>TSE (RTC_TS input enable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS input floating</td>
<td>00 0 0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 0 1</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 0 0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 0 1</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 0 0</td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00 0 1</td>
<td>1</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Don’t care</td>
<td>Don’t care</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.
49.3.4 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and non-privileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:
- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and non-privileged modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRIV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and non-privileged modes.
- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privileged mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and non-privileged mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR can be read and written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE, ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in

<table>
<thead>
<tr>
<th>Table 505. RTC_OUT mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSEL[1:0] bits ALARM output enable</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>00</td>
</tr>
<tr>
<td>01 or 10 or 11</td>
</tr>
</tbody>
</table>
RTC_SR, and ALRBMF in RTC_MISR can be read and written only in privilege mode.

- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR can be read and written only in privilege mode.

- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR can be read and written only in privilege mode.

A non-privileged access to a privileged-protected register is denied:
- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

49.3.5 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see Figure 676: RTC block diagram):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

Ck_apre is given by the following formula:

\[
f_{\text{CK APRE}} = \frac{f_{\text{RTCCLK}}}{\text{PREDIV}_A + 1}
\]

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.
The ck_spref clock can be used either to update the calendar or as timebase for the 16-bit
wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload
timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous
prescaler (see Section 49.3.9: Periodic auto-wake-up for details).

**Binary mode (BIN=01)**

The SSR binary down-counter is extended to 32-bit length and is free running. The time and
date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler
configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don’t care.

**Mixed mode (BIN=10 or 11)**

The SSR binary down-counter is extended to 32-bit length and is free running. The time and
date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler
configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are
used to define when the calendar is incremented by 1 second, using the SSR least
significant bits.

### 49.3.6 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which
are synchronized with PCLK (APB clock). They can also be accessed directly in order to
avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and
the RSF bit of RTC_ICSR register is set (see Section 49.6.11: RTC shift control register
(RTC_SHIFTR)). The copy is not performed in Stop and Standby mode. When exiting these
modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow
registers. It is possible to make a direct access to the calendar registers by setting the
BYPRESHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user
accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPRESHAD = 0 mode, the
frequency of the APB clock (fAPB) must be at least 7 times the frequency of the RTC clock
(fRTCCLK).

The shadow registers are reset by system reset.
49.3.7 **Calendar ultra-low power mode**

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to Section: RTC ultra-low-power mode).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

49.3.8 **Programmable alarms**

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

**Caution:** If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPAiLM output. TAMPAiLM output polarity can be configured through bit POL the RTC_CR register.

49.3.9 **Periodic auto-wake-up**

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.
The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCCLK) divided by 2, 4, 8, or 16.
  
  When RTCCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.

- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.
  
  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case $2^{16}$ is added to the 16-bit counter current value. When the initialization sequence is complete (see Programming the wake-up timer on page 2155), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section Section 49.3.1: RTC block diagram).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop, and Standby) have no influence on the wake-up timer.

49.3.10 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.
RTC register write protection

After system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable RTC registers write access.

After Backup domain reset, some of the RTC registers are write-protected: RTC_TR, RTC_DR, RTC_PRER, RTC_CALR, RTC_SHIFTR, the bits INIT, BIN and BCDU in RTC_ICSR and the bits FMT, SUB1H, ADD1H, REFCKON in RTC_CR.

The following steps are required to unlock the write protection on the protected RTC registers.

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Writing a wrong key reactivates the write protection.

The protection mechanism is not affected by system reset.

The registers protected by INITPRIV are write-protected by the INIT KEY.

The registers protected by CALPRIV are write-protected by the CAL KEY.

In case PRIV or INITPRIV is set in the RTC_PRIVCFGR: the INIT KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege mode defined by PRIV, INITPRIV configuration.

In case PRIV or CALPRIV is set in the RTC_PRIVCFGR: the CAL KEY is unlocked and locked only if the write accesses into the RTC_WPR register are done in the privilege mode defined by PRIV, CALPRIV configuration.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   - If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   - If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   - If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   - If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.
When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 μs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

**Note:** After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

**Note:** To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

**Daylight saving time**

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

**Programming the alarm**

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

**Note:** Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

**Programming the wake-up timer**

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):
1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed. This step must be skipped in calendar initialization mode.
   - If WUCKSEL[2] = 0: WUTFW is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTFW is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

49.3.11 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (fPCLK) must be equal to or greater than seven times the RTC clock frequency (fRTCCLK). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to Calendar initialization and configuration on page 2154): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.
After synchronization (refer to Section 49.3.13: RTC synchronization): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

**When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)**

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the results to confirm that the data is coherent and correct. Alternatively, the software can just compare the two results of the least-significant calendar register.

*Note:* While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

### 49.3.12 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

### 49.3.13 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.
If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

**Caution:** In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

**Caution:** Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

**Caution:** This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

### 49.3.14 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:
- PREDIV_A = 0x007F
- PREDIV_S = 0x00FF

**Note:** RTC_REFIN clock detection is not available in Standby mode.
49.3.15 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual \(ck_{\text{cal}}\) pulses).

If \(LPCAL=0\): \(ck_{\text{cal}} = \text{RTCCLK}\)

If \(LPCAL=1\): \(ck_{\text{cal}} = \text{ck}_{\text{apre}}\)

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTT ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on \(ck_{\text{apre}}\) instead of \(\text{RTCCLK}\). The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about \(2^{20} \times \text{PREDIV}_A \times \text{RTCCLK}\) pulses instead of \(2^{20}\) \(\text{RTCCLK}\) pulses when \(LPCAL=0\).

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of \(ck_{\text{cal}}\) clock cycles to be masked during the calibration cycle:

- Setting the bit \(\text{CALM}[0]\) to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting \(\text{CALM}[1]\) to 1 causes two additional cycles to be masked.
- Setting \(\text{CALM}[2]\) to 1 causes four additional cycles to be masked.
- and so on up to \(\text{CALM}[8]\) set to 1 which causes 256 clocks to be masked.

Note: \(\text{CALM}[8:0]\) (RTC_CALR) specifies the number of \(ck_{\text{cal}}\) pulses to be masked during the calibration cycle. Setting the bit \(\text{CALM}[0]\) to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when \(\text{cal}_{\text{cnt}}[19:0]\) is 0x80000; \(\text{CALM}[1]\) = 1 causes two other cycles to be masked (when \(\text{cal}_{\text{cnt}}\) is 0x40000 and 0xC0000); \(\text{CALM}[2]\) = 1 causes four other cycles to be masked (\(\text{cal}_{\text{cnt}}\) = 0x20000/0x60000/0xA0000/0xE0000); and so on up to \(\text{CALM}[8]\) = 1 which causes 256 clocks to be masked (\(\text{cal}_{\text{cnt}}\) = 0xXX800).

While \(\text{CALM}\) permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit \(\text{CALP}\) can be used to increase the frequency by 488.5 ppm. Setting \(\text{CALP}\) to 1 effectively inserts an extra \(ck_{\text{cal}}\) pulse every \(2^{20}\) \(ck_{\text{cal}}\) cycles, which means that 512 clocks are added during every calibration cycle.

Using \(\text{CALM}\) together with \(\text{CALP}\), an offset ranging from -511 to +512 \(ck_{\text{cal}}\) cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency \(F_{\text{CAL}}\) given the input frequency \(F_{\text{RTCCLK}}\) is as follows:

\[
F_{\text{CAL}} = F_{\text{RTCCLK}} \times \left[1 + \left(\text{CALP} \times 512 - \text{CALM}\right) / \left(2^{20} + \text{CALM} - \text{CALP} \times 512\right)\right]
\]

Caution: \(\text{PREDIV}_A\) must be greater or equal to 3.

Calibration when \(\text{PREDIV}_A < 3\)

The \(\text{CALP}\) bit cannot be set to 1 when the asynchronous prescaler value (\(\text{PREDIV}_A\) bits in RTC_PRER register) is less than 3. If \(\text{CALP}\) was already set to 1 and \(\text{PREDIV}_A\) bits are
set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

\[ F_{\text{CAL}} = F_{\text{RTCCLK}} \times \left[ 1 + \frac{256 - \text{CALM}}{2^{20} + \text{CALM} - 256} \right] \]

In this case, \( \text{CALM}[7:0] \) equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

**Verifying the RTC calibration**

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).

- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.

- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.
**Re-calibration on-the-fly**

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1.
3. Within three \( \text{ck}_\text{apre} \) cycles after the write operation to RTC_CALR, the new calibration settings take effect.

**49.3.16 Timestamp function**

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to RTC control register (RTC_CR) and refer to Section : Timestamp on tamper event.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the \( V_{\text{BAT}} \) supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

**Note:**

TSF is set up to 2 \( \text{ck}_\text{apre} \) cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs due to synchronization process. TSF is set up to 3 \( \text{ck}_\text{apre} \) cycles after tamper flags.

TSOVF is set up to only 1 \( \text{ck}_\text{apre} \) cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as ‘1’ while TSF is still ‘0’. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

**Caution:**

If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

**49.3.17 Calibration clock output**

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.
If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is \( f_{RTCCLK} / 64 \). This corresponds to a calibration output at 512 Hz for an RTCCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is \( f_{RTCCLK} / (256 \times (PREDIV_A+1)) \). This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = 0x7F, PREDIV_S = 0xFF), with an RTCCCLK frequency at 32.768 kHz.

**Note:** When COSEL is cleared, the CALIB output is the output of the 6th stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20. When COSEL is set, the CALIB output is the output of the 8th stage of the synchronous prescaler.

### 49.3.18 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tamper flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

**TAMPALRM output**

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

**Note:** Once the TAMPALRM output is enabled, it has priority over CALIB on RTC_OUT1. In case the TAMPALRM is configured open-drain in the RTC, the RTC_OUT1 GPIO must be configured as input.

### 49.4 RTC low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. RTC interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.</td>
</tr>
</tbody>
</table>
The table below summarizes the RTC pins and functions capability in all modes.

### Table 507. RTC pins functionality over modes

<table>
<thead>
<tr>
<th>Functions</th>
<th>Functional in all low-power modes except Standby mode</th>
<th>Functional in Standby mode</th>
<th>Functional in V_{BAT} mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC_TS</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTC_REFIN</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>RTC_OUT1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>RTC_OUT2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

#### 49.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register. The interrupt output is also activated.

### Table 508. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit(2)</th>
<th>Interrupt clear method</th>
<th>Exit from low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTC</td>
<td>Alarm A</td>
<td>ALRAF</td>
<td>ALRAIE</td>
<td>write 1 in CALRAF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Alarm B</td>
<td>ALRBFW</td>
<td>ALRBIE</td>
<td>write 1 in CALRBFW</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Timestamp</td>
<td>TSF</td>
<td>TSIE</td>
<td>write 1 in CTSF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Wake-up timer</td>
<td>WUTF</td>
<td>WUTIE</td>
<td>write 1 in CWUTF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>SSR underflow (reload)</td>
<td>SSRUF</td>
<td>SSRIE</td>
<td>write 1 in CSSRUF</td>
<td>Yes(3)</td>
</tr>
</tbody>
</table>

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

#### 49.6 RTC registers

Refer to Section 1.2 of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).
49.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 2154 and Reading the calendar on page 2156.

This register is write protected. The write access procedure is described in RTC register write protection on page 2154.

This register can be write-protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x00
Backup domain reset value: 0x0000 0000
System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PM</td>
<td>HT[1:0]</td>
<td>HU[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format
49.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to Calendar initialization and configuration on page 2154 and Reading the calendar on page 2156.

This register is write protected. The write access procedure is described in RTC register write protection on page 2154.

This register can be write-protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units

000: forbidden

001: Monday

...

111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

**Note:** The calendar is frozen when reaching the maximum value, and can’t roll over.
49.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08
Backup domain reset value: 0x0000 0000
System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| SS[15:0] |
| r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:0 SS[31:0]: Synchronous binary counter
- SS[31:16]: Synchronous binary counter MSB values
  - When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):
    - SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.
  - When BCD mode is selected (BIN=00):
    - SS[31:16] are forced by hardware to 0x0000.
- SS[15:0]: Subsecond value/synchronous binary counter LSB values
  - When Binary mode is selected (BIN = 01 or 10 or 11):
    - SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.
  - When BCD mode is selected (BIN=00):
    - SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:
      - Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)
      - SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

49.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in RTC register write protection on page 2154.

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x0C
Backup domain reset value: 0x0000 0007
System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BCDU[2:0]</td>
<td>BIN[1:0]</td>
<td>INIT</td>
<td>INITF</td>
<td>RSF</td>
<td>INITS</td>
<td>SHPF</td>
<td>WUTW</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rc_w0</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

- 0x0: 1s calendar increment is generated each time SS[7:0] = 0
- 0x1: 1s calendar increment is generated each time SS[8:0] = 0
- 0x2: 1s calendar increment is generated each time SS[9:0] = 0
- 0x3: 1s calendar increment is generated each time SS[10:0] = 0
- 0x4: 1s calendar increment is generated each time SS[11:0] = 0
- 0x5: 1s calendar increment is generated each time SS[12:0] = 0
- 0x6: 1s calendar increment is generated each time SS[13:0] = 0
- 0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

- 00: Free running BCD calendar mode (Binary mode disabled).
- 01: Free running Binary mode (BCD mode disabled)
- 10: Free running BCD calendar and Binary modes
- 11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

- 0: Free running mode
- 1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

- 0: Calendar registers update is not allowed
- 1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSSHAD = 1). This bit can also be cleared by software.

- 0: Calendar shadow registers not yet synchronized
- 1: Calendar shadow registers synchronized

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

- 0: Calendar has not been initialized
- 1: Calendar has been initialized
Bit 3  **SHPF**: Shift operation pending
   - This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.
   - 0: No shift operation is pending
   - 1: A shift operation is pending

Bit 2  **WUTWF**: Wake-up timer write flag
   - This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.
   - It is cleared by hardware in initialization mode.
   - 0: Wake-up timer configuration update not allowed except in initialization mode
   - 1: Wake-up timer configuration update allowed

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

### 49.6.5  RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to [Calendar initialization and configuration on page 2154](#).

This register is write protected. The write access procedure is described in [RTC register write protection on page 2154](#).

This register can be write-protected against non-privileged access. Refer to [Section 49.3.4: RTC privilege protection modes](#).

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>PREDIV_A[6:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

**Bits 22:16**  **PREDIV_A[6:0]**: Asynchronous prescaler factor
   - This is the asynchronous division factor:
     \[ \text{ck_apre frequency} = \frac{\text{RTCCLK frequency}}{\text{PREDIV_A}+1} \]
   - Bit 15  Reserved, must be kept at reset value.

**Bits 14:0**  **PREDIV_S[14:0]**: Synchronous prescaler factor
   - This is the synchronous division factor:
     \[ \text{ck_spre frequency} = \frac{\text{ck_apre frequency}}{\text{PREDIV_S}+1} \]
49.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.
This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x14
Backup domain reset value: 0x0000 FFFF
System reset: not affected

<table>
<thead>
<tr>
<th>WUTOCLR[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WUT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bits 31:16 **WUTOCLR[15:0]:** Wake-up auto-reload output clear value
When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].
When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]:** Wake-up auto-reload value bits
When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.
When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.
The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

49.6.7 RTC control register (RTC_CR)

*This register is write protected. The write access procedure is described in RTC register write protection on page 2154.*
This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>OUT2EN: RTC_OUT2 output enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:</td>
</tr>
<tr>
<td></td>
<td><strong>OUT2EN = 0</strong>: RTC output 2 disable</td>
</tr>
<tr>
<td></td>
<td>If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1</td>
</tr>
<tr>
<td></td>
<td>If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1</td>
</tr>
<tr>
<td></td>
<td><strong>OUT2EN = 1</strong>: RTC output 2 enable</td>
</tr>
<tr>
<td></td>
<td>If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2</td>
</tr>
<tr>
<td></td>
<td>If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2</td>
</tr>
<tr>
<td></td>
<td>If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>TAMPALRM_TYPE: TAMPALRM output type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TAMPALRM is push-pull output</td>
</tr>
<tr>
<td>1</td>
<td>TAMPALRM is open-drain output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>TAMPALRM_PU: TAMPALRM pull-up enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No pull-up is applied on TAMPALRM output</td>
</tr>
<tr>
<td>1</td>
<td>A pull-up is applied on TAMPALRM output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>ALRBFCLR: Alarm B flag automatic clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.</td>
</tr>
<tr>
<td>1</td>
<td>Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 27</th>
<th>ALRAFCLR: Alarm A flag automatic clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.</td>
</tr>
<tr>
<td>1</td>
<td>Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 26</th>
<th>TAMPOE: Tamper detection output enable on TAMPALRM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The tamper flag is not routed on TAMPALRM</td>
</tr>
<tr>
<td>1</td>
<td>The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.</td>
</tr>
</tbody>
</table>
Bit 25 **TAMPTS**: Activate timestamp on tamper detection event

- 0: Tamper detection event does not cause a RTC timestamp to be saved
- 1: Save RTC timestamp on tamper detection event

TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.

*Note: TAMPTS must be cleared before entering RTC initialization mode.*

Bit 24 **ITSE**: timestamp on internal event enable

- 0: internal event timestamp disabled
- 1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable

- 0: Calibration output disabled
- 1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection

- 00: Output disabled
- 01: Alarm A output enabled
- 10: Alarm B output enabled
- 11: Wake-up output enabled

Bit 20 **POL**: Output polarity

- 0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxXF/ITAMPxXF is asserted (if TAMPOE = 1).
- 1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxXF/ITAMPxXF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection

- 0: Calibration output is 512 Hz
- 1: Calibration output is 1 Hz

These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 49.3.17: Calibration clock output*.

Bit 18 **BKP**: Backup

- This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)

- When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
- Setting this bit has no effect when current hour is 0.
- 0: No effect
- 1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)

- When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
- 0: No effect
- 1: Adds 1 hour to the current time. This can be used for summer time change

Bit 15 **TSIE**: Timestamp interrupt enable

- 0: Timestamp interrupt disable
- 1: Timestamp interrupt enable
<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td><strong>WUTIE</strong>: Wake-up timer interrupt enable</td>
<td>Wake-up timer interrupt disabled</td>
<td>Wake-up timer interrupt enabled</td>
</tr>
<tr>
<td>13</td>
<td><strong>ALRBIIE</strong>: Alarm B interrupt enable</td>
<td>Alarm B interrupt disabled</td>
<td>Alarm B interrupt enabled</td>
</tr>
<tr>
<td>12</td>
<td><strong>ALRAIE</strong>: Alarm A interrupt enable</td>
<td>Alarm A interrupt disabled</td>
<td>Alarm A interrupt enabled</td>
</tr>
<tr>
<td>11</td>
<td><strong>TSE</strong>: timestamp enable</td>
<td>timestamp disable</td>
<td>timestamp enable</td>
</tr>
<tr>
<td>10</td>
<td><strong>WUTE</strong>: Wake-up timer enable</td>
<td>Wake-up timer disabled</td>
<td>Wake-up timer enabled</td>
</tr>
</tbody>
</table>

*Note: When the wake-up timer is disabled, wait for WUTF = 1 before enabling it again.*

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td><strong>ALRBE</strong>: Alarm B enable</td>
<td>Alarm B disabled</td>
<td>Alarm B enabled</td>
</tr>
<tr>
<td>8</td>
<td><strong>ALRAE</strong>: Alarm A enable</td>
<td>Alarm A disabled</td>
<td>Alarm A enabled</td>
</tr>
<tr>
<td>7</td>
<td><strong>SSRUIE</strong>: SSR underflow interrupt enable</td>
<td>SSR underflow interrupt disabled</td>
<td>SSR underflow interrupt enabled</td>
</tr>
<tr>
<td>6</td>
<td><strong>FMT</strong>: Hour format</td>
<td>24 hour/day format</td>
<td>AM/PM hour format</td>
</tr>
<tr>
<td>5</td>
<td><strong>BYPHAD</strong>: Bypass the shadow registers</td>
<td>Calendar values taken from shadow registers, updated every two RTCCLK cycles</td>
<td>Calendar values taken directly from calendar counters</td>
</tr>
</tbody>
</table>

*Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPHAD must be set to 1.*
Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled

**Note**: **BIN must be 0x00 and PREDIV_S must be 0x00FF.**

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event

TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, $2^{16}$ is added to the WUT counter value.

**Note**: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

$WUT = \text{wake-up unit counter value. } WUT = (0x0000 \text{ to } 0xFFFF) + 0x10000 \text{ added when } WUCKSEL[2:1 = 11].$

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

### 49.6.8 RTC privilege mode control register (RTC_PRIVCFGFR)

This register can be written only when the APB access is privileged.

Address offset: 0x1C
Backup domain reset value: 0x0000 0000
System reset: not affected

```plaintext
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Bits 31:16 Reserved, must be kept at reset value.

Bit 15 PRIV: RTC privilege protection
0: All RTC registers can be written when the APB access is privileged or non-privileged, except the registers protected by other privilege protection bits.
1: All RTC registers can be written only when the APB access is privileged.

Bit 14 INITPRIV: Initialization privilege protection
0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or non-privileged.
1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 CALPRIV: Shift register, Delight saving, calibration and reference clock privilege protection
0: Shift register, Delight saving, calibration and reference clock can be written when the APB access is privileged or non-privileged.
1: Shift register, Delight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 TSPRIV: Timestamp privilege protection
0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 WUTPRIV: Wake-up timer privilege protection
0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 ALRBPRIV: Alarm B privilege protection
0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 ALRAPRIV: Alarm A and SSR underflow privilege protection
0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or non-privileged.
1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 49.3.4: RTC privilege protection modes for details on the read protection.
This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the RTC_PRIVCFGR configuration (refer to Section 49.3.4: RTC privilege protection modes).

### 49.6.9 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to RTC register write protection for a description of how to unlock RTC register write protection.

### 49.6.10 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in RTC register write protection on page 2154.

This register can be write-protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CALP</th>
<th>CALW8</th>
<th>CALW16</th>
<th>LPCAL</th>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>CALM[8:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:16  Reserved, must be kept at reset value.

Bit 15  **CALP**: Increase frequency of RTC by 488.5 ppm

0: No RTCCCLK pulses are added.

1: One RTCCCLK pulse is effectively inserted every \(2^{11}\) pulses (frequency increased by 488.5 ppm).

This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. If the input frequency is 32768 Hz, the number of RTCCCLK pulses added during a 32-second window is calculated as follows:

\[512 \times \text{CALP} - \text{CALM}\]

Refer to Section 49.3.15: RTC smooth digital calibration.

Bit 14  **CALW8**: Use an 8-second calibration cycle period

When CALW8 is set to 1, the 8-second calibration cycle period is selected.

Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 49.3.15: RTC smooth digital calibration.

Bit 13  **CALW16**: Use a 16-second calibration cycle period

When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.

Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 49.3.15: RTC smooth digital calibration.

Bit 12  **LPCAL**: RTC low-power mode

0: Calibration window is \(2^{20}\) RTCCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is \(2^{20}\) ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9  Reserved, must be kept at reset value.

Bits 8:0  **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of \(2^{20}\) RTCCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See Section 49.3.15: RTC smooth digital calibration on page 2159.
49.6.11 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in RTC register write protection on page 2154.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>ADD1S</th>
<th>SUBFS[14:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>w</td>
<td>w w w w w w w w w w w w w w</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 ADD1S: Add one second
0: No effect
1: Add one second to the clock/calendar
This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).
This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 SUBFS[14:0]: Subtract a fraction of a second
These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).
The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:
Delay (seconds) = SUBFS / (PREDIV_S + 1)
A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:
Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).
In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.
Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.
49.6.12  RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x30
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PM</td>
<td>H[T:1]</td>
<td>H[U:3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M[N:2]</td>
<td>M[NU:3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S[T:3]</td>
<td>S[U:3]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 22  **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20  **HT[1:0]**: Hour tens in BCD format.

Bits 19:16  **H[U:3]**: Hour units in BCD format.

Bit 15  Reserved, must be kept at reset value.

Bits 14:12  **M[N:2]**: Minute tens in BCD format.

Bits 11:8  **M[NU:3]**: Minute units in BCD format.

Bit 7  Reserved, must be kept at reset value.

Bits 6:4  **S[T:3]**: Second tens in BCD format.

Bits 3:0  **S[U:3]**: Second units in BCD format.
49.6.13 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x34
Backup domain reset value: 0x0000 0000
System reset: not affected

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Bits 31:16  Reserved, must be kept at reset value.
Bits 15:13  WDU[2:0]: Week day units
Bit 12    MT: Month tens in BCD format
Bits 11:8  MU[3:0]: Month units in BCD format
Bits 7:6   Reserved, must be kept at reset value.
Bits 5:4   DT[1:0]: Date tens in BCD format
Bits 3:0   DU[3:0]: Date units in BCD format

49.6.14 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x38
Backup domain reset value: 0x0000 0000
System reset: not affected

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SS[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SS[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>
```
Bits 31:0  SS[31:0]: Subsecond value/synchronous binary counter values
SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

49.6.15  RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x40
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  MSK4: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30  WDSEL: Weekday selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28  DT[1:0]: Date tens in BCD format
Bits 27:24  DU[3:0]: Date units or day in BCD format

Bit 23  MSK3: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22  PM: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20  HT[1:0]: Hour tens in BCD format
Bits 19:16  HU[3:0]: Hour units in BCD format

Bit 15  MSK2: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12  MNT[2:0]: Minute tens in BCD format
Bits 11:8  MNU[3:0]: Minute units in BCD format
49.6.16 **RTC alarm A subsecond register (RTC_ALRMASSR)**

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to *Section 49.3.4: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one clock cycle after reaching RTC_ALRABINR.SS[31:0].

*Note*: **SSCLR** must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 **Reserved**, must be kept at reset value.
49.6.17 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bits 29:24</th>
<th>MASKSS[5:0]: Mask the most-significant bits starting at this bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).</td>
<td></td>
</tr>
<tr>
<td>1: SS[31:1] are don’t care in Alarm A comparison. Only SS[0] is compared.</td>
<td></td>
</tr>
<tr>
<td>2: SS[31:2] are don’t care in Alarm A comparison. Only SS[1:0] are compared.</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>31: SS[31] is don’t care in Alarm A comparison. Only SS[30:0] are compared.</td>
<td></td>
</tr>
<tr>
<td>From 32 to 63: All 32 SS bits are compared and must match to activate alarm.</td>
<td></td>
</tr>
</tbody>
</table>

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

| Bits 23:15 | Reserved, must be kept at reset value. |

<table>
<thead>
<tr>
<th>Bits 14:0</th>
<th>SS[14:0]: Subseconds value</th>
</tr>
</thead>
<tbody>
<tr>
<td>This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.</td>
<td></td>
</tr>
<tr>
<td>This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.</td>
<td></td>
</tr>
</tbody>
</table>

Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMPACTR1.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>MSK4: Alarm B date mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Alarm B set if the date and day match</td>
<td></td>
</tr>
<tr>
<td>1: Date and day don’t care in alarm B comparison</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>WDS</th>
<th>Week day selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: DU[3:0] represents the date units</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: DU[3:0] represents the week day. DT[1:0] is don’t care.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bits 29:28 | DT[1:0]: Date tens in BCD format |
| Bits 27:24 | DU[3:0]: Date units or day in BCD format |
Bit 23 **MSK3**: Alarm B hours mask
   0: Alarm B set if the hours match
   1: Hours don’t care in alarm B comparison

Bit 22 **PM**: AM/PM notation
   0: AM or 24-hour format
   1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
   0: Alarm B set if the minutes match
   1: Minutes don’t care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
   0: Alarm B set if the seconds match
   1: Seconds don’t care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

49.6.18 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected
Bit 31  **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one
ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].

*Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).*

Bit 30  Reserved, must be kept at reset value.

Bits 29:24  **MASKSS[5:0]**: Mask the most-significant bits starting at this bit
0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is
incremented (assuming that the rest of the fields match).
1: SS[31:1] are don’t care in Alarm B comparison. Only SS[0] is compared.
2: SS[31:2] are don’t care in Alarm B comparison. Only SS[1:0] are compared.
...
31: SS[31] is don’t care in Alarm B comparison. Only SS[30:0] are compared.
From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

*Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never
compared. This bit can be different from 0 only after a shift operation.*

Bits 23:15  Reserved, must be kept at reset value.

Bits 14:0  **SS[14:0]**: Subseconds value
This value is compared with the contents of the synchronous prescaler counter to determine
if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.
This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written
through RTC_ALRBBINR.

*Note: SS[3:0] must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMPERCLR.*

### 49.6.19 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually
protected against non-privileged access. Refer to *Section 49.3.4: RTC privilege protection
modes.*

Address offset: 0x50
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag
This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0.
SSRUF is not set when SSCLR = 1.

*Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload.*

Bit 5 **ITSF**: Internal timestamp flag
This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag
This flag is set by hardware when a timestamp event occurs while TSF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag
This flag is set by hardware when a timestamp event occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

*Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.*

Bit 2 **WUTF**: Wake-up timer flag
This flag is set by hardware when the wake-up auto-reload counter reaches 0.
If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.
If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag
This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

*Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.*
### 49.6.20 RTC masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to *Section 49.3.4: RTC privilege protection modes*.

- **Address offset:** 0x54
- **Backup domain reset value:** 0x0000 0000
- **System reset:** not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

- **Bit 6 SSRUMF:** SSR underflow masked flag
  This flag is set by hardware when the SSR underflow interrupt occurs.

- **Bit 5 ITSMF:** Internal timestamp masked flag
  This flag is set by hardware when a timestamp on the internal event occurs and timestampinterrupt is raised.

- **Bit 4 TSOVMF:** Timestamp overflow masked flag
  This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
  It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

- **Bit 3 TSMF:** Timestamp masked flag
  This flag is set by hardware when a timestamp interrupt occurs.
  If ITSF flag is set, TSF must be cleared together with ITSF.

- **Bit 2 WUTMF:** Wake-up timer masked flag
  This flag is set by hardware when the wake-up timer interrupt occurs.
  This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

- **Bit 1 ALRBMF:** Alarm B masked flag
  This flag is set by hardware when the alarm B interrupt occurs.

- **Bit 0 ALRAMF:** Alarm A masked flag
  This flag is set by hardware when the alarm A interrupt occurs.

*Note:* The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.
This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

### 49.6.21 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

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

Bits 31:7 Reserved, must be kept at reset value.

- **Bit 6 CSSRUF**: Clear SSR underflow flag
  - Writing '1' in this bit clears the SSRUF in the RTC_SR register.

- **Bit 5 CITSF**: Clear internal timestamp flag
  - Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

- **Bit 4 CTSOVF**: Clear timestamp overflow flag
  - Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
  - It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

- **Bit 3 CTSF**: Clear timestamp flag
  - Writing 1 in this bit clears the TSF bit in the RTC_SR register.
  - If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

- **Bit 2 CWUTF**: Clear wake-up timer flag
  - Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

- **Bit 1 CALRBF**: Clear alarm B flag
  - Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

- **Bit 0 CALRAF**: Clear alarm A flag
  - Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.
49.6.22  RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x70
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>SS[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>SS[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSRR, and so can also be read or written through RTC_ALRMASSR.


49.6.23  RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register, or in initialization mode.

This register can be protected against non-privileged access. Refer to Section 49.3.4: RTC privilege protection modes.

Address offset: 0x74
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>SS[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>SS[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:0 **SS[31:0]**: Synchronous counter alarm value in Binary mode

This value is compared with the contents of the synchronous counter to determine if Alarm Bit is to be activated. Only bits 0 up MASKSS-1 are compared.

**SS[14:0]** is the mirror of **SS[14:0]** in the RTC_ALRMBSSRR, and so can also be read or written through RTC_ALRMBSSR.

*Note:* **SS[3:0]** must be 0000 when SSCLR is set with ATCKSEL[3] = 1 in TAMP_ATCR1.
### 49.6.24 RTC register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>WUTOCLR[15:0]</th>
<th>WUT[15:0]</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>RTC_TR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x04</td>
<td>RTC_DR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08</td>
<td>RTC_SSR</td>
<td>SS[31:16]</td>
<td>SS[15:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C</td>
<td>RTC_ICSR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x10</td>
<td>RTC_PRER</td>
<td></td>
<td></td>
<td>1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x14</td>
<td>RTC_WUTR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18</td>
<td>RTC_CR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x1C</td>
<td>RTC_PRIVCFGR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x24</td>
<td>RTC_WPR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x28</td>
<td>RTC_CALR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x2C</td>
<td>RTC_SHIFTR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x30</td>
<td>RTC_TSTR</td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Real-time clock (RTC) RM0477
Refer to Section 2.3 for the register boundary addresses.
50 Tamper and backup registers (TAMP)

50.1 Introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in $V_{BAT}$ mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 8 tamper pins and 11 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

50.2 TAMP main features

- A tamper detection can optionally erase the backup registers, backup SRAM, and cryptographic peripherals. The device resources protected by tamper are named "device secrets".
- 32 32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by $V_{BAT}$ when the $V_{DD}$ power is switched off.
- Up to 8 tamper pins for 8 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 11 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.
50.3  TAMP functional description

50.3.1  TAMP block diagram

Figure 677. TAMP block diagram

1. The number of external and internal tampers depends on products.
50.3.2 TAMP pins and internal signals

Table 510. TAMP input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_INx (x = pin index)</td>
<td>Input</td>
<td>Tamper input pin</td>
</tr>
<tr>
<td>TAMP_OUTx (x = pin index)</td>
<td>Output</td>
<td>Tamper output pin (active mode only)</td>
</tr>
</tbody>
</table>

Table 511. TAMP internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_ker_ck</td>
<td>Input</td>
<td>TAMP kernel clock, connected to rtc_ker_ck and also named RTCCCLK in this document</td>
</tr>
<tr>
<td>tamp_pclk</td>
<td>Input</td>
<td>TAMP APB clock, connected to rtc_pclk</td>
</tr>
<tr>
<td>tamp_itamp[y]</td>
<td>Inputs</td>
<td>Internal tamper event sources</td>
</tr>
<tr>
<td>tamp_evt</td>
<td>Output</td>
<td>Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.</td>
</tr>
<tr>
<td>tamp_potential</td>
<td>Output</td>
<td>Potential tamper detection signal, used for device secrets(^{(1)}) protection. This signal is active when: -- a tamper event detection flag (internal or external tamper), is generated in potential mode. -- or a software request is done by writing BKBLOCK to 1</td>
</tr>
<tr>
<td>tamp_confirmed</td>
<td>Output</td>
<td>Confirmed tamper detection signal, used for device secrets(^{(1)}) protection. This signal is active when: -- a tamper event detection flag (internal or external tamper), is generated in confirmed mode. -- or a software request is done by writing BKERASE to 1</td>
</tr>
<tr>
<td>tamp_potential_rpcfgz</td>
<td>Output</td>
<td>Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: -- a tamper event detection flag (internal or external tamper), is generated in potential mode. -- or a software request is done by writing BKBLOCK to 1</td>
</tr>
</tbody>
</table>
The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some detections modes are not available in some low-power modes or VBAT depending on the selected clock (refer to Section 50.4: TAMP low-power modes for more details).

### Table 511. TAMP internal input/output signals (continued)

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_confirmed_rpcfgz (z = signal index)</td>
<td>Output</td>
<td>Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1</td>
</tr>
<tr>
<td>tamp_it</td>
<td>Output</td>
<td>TAMP interrupt (refer to Section 50.5: TAMP interrupts for details)</td>
</tr>
<tr>
<td>tamp_trg[x] (x = signal index)</td>
<td>Output</td>
<td>Tamper detection trigger</td>
</tr>
<tr>
<td>tamp_bhk</td>
<td>Output</td>
<td>Tamper boot hardware key bus</td>
</tr>
</tbody>
</table>

1. Refer to Table 512: TAMP interconnection.

The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter:
- backup registers
- RHUK (root hardware unique key) in system Flash memory and BHK (boot hardware key) hardware buses to SAES are blocked.

The tamp_potential signal is used to erase the device secrets listed hereafter:
- SAES, AES, HASH peripherals
- PKA SRAM

The device secrets access is blocked when erase is on-going.

### Table 512. TAMP interconnection

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_evt</td>
<td>rtc_tamp_evt used to generate a timestamp event</td>
</tr>
<tr>
<td>tamp_potential</td>
<td>The tamp_potential signal is used to block the read and write accesses to the device secrets listed hereafter: – backup registers – RHUK (root hardware unique key) in system Flash memory and BHK (boot hardware key) hardware buses to SAES are blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is on-going.</td>
</tr>
<tr>
<td>tamp_confirmed</td>
<td>The tamp_confirmed signal is used to erase the device secrets listed hereafter: – backup registers – MCE keys and CRC registers – SAES, AES, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is on-going. RHUK in system Flash memory (root hardware unique key) hardware bus to SAES is blocked.</td>
</tr>
</tbody>
</table>
### 50.3.3 GPIOs controlled by the RTC and TAMP

Refer to Section 49.3.3: GPIOs controlled by the RTC and TAMP.

### 50.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

---

**Table 512. TAMP interconnection (continued)**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Source/Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>tamp_potential_rpcfg0</td>
<td>When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg0 signal is used to block the read and write accesses to the device secrets listed hereafter:</td>
</tr>
<tr>
<td></td>
<td>– Backup SRAM</td>
</tr>
<tr>
<td>tamp_confirmed_rpcfg0</td>
<td>When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg0 signal is used to erase the device secrets listed hereafter:</td>
</tr>
<tr>
<td></td>
<td>– Backup SRAM</td>
</tr>
<tr>
<td></td>
<td>The device secrets access is blocked when erase is on-going.</td>
</tr>
<tr>
<td>tamp_itamp1</td>
<td>Backup domain voltage threshold monitoring[^1]</td>
</tr>
<tr>
<td>tamp_itamp2</td>
<td>Temperature monitoring[^1]</td>
</tr>
<tr>
<td>tamp_itamp3</td>
<td>LSE monitoring (LSECSS)[^2]</td>
</tr>
<tr>
<td>tamp_itamp4</td>
<td>HSE monitoring (rcc_hsecss_fail)</td>
</tr>
<tr>
<td>tamp_itamp5</td>
<td>RTC calendar overflow (rtc_calovf)</td>
</tr>
<tr>
<td>tamp_itamp6</td>
<td>JTAG/SWD access when NVSTATE ≠ OPEN</td>
</tr>
<tr>
<td>tamp_itamp7</td>
<td>ADC2 watchdog monitoring 1</td>
</tr>
<tr>
<td>tamp_itamp8[^3]</td>
<td>Monotonic counter 1 overflow</td>
</tr>
<tr>
<td>tamp_itamp9</td>
<td>Cryptographic peripherals fault (SAES or AES or PKA or TRNG)</td>
</tr>
<tr>
<td>tamp_itamp11</td>
<td>IWDG reset when tamper flag is set (potential tamper timeout)</td>
</tr>
<tr>
<td>tamp_itamp15</td>
<td>System fault</td>
</tr>
<tr>
<td>tamp_bhk</td>
<td>saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.</td>
</tr>
</tbody>
</table>

[^1]: This monitoring must be enabled by setting MONEN in PWR control status register 1 (PWR_CSR1).
[^2]: This monitoring must be enabled by setting LSECSSON in RCC Backup domain control register (RCC_BDCR).
[^3]: This signal is generated in the TAMP peripheral.
50.3.5 Backup registers protection zones

The backup registers protection is configured thanks to BKPRW[7:0] and BKPW[7:0] (refer to the figure below):

![Backup registers protection zones](image)

1. l = last backup register index

50.3.6 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and non-privileged modes, except for the TAMP privilege configuration register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
- Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counters which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode.
- Reading TAMP_CFGR, TAMP_PRIVCFGR is always possible in privilege and non-privileged modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counters which have their own protection setting.

The backup registers protection is configured thanks to BKPRW[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRW[7:0], BKPW[7:0] and BKPWPRIV for the protection zone 2 (refer to Figure 678). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

A non-privileged access to a privileged-protected register is denied:
- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

50.3.7 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.
Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_CFGR register. Once BHKLOCK is set, the 8 backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a tamper event or when the product state is opened. It is also cleared with BKERASE command (in all cases the backup registers are also erased).

Refer to section secure AES co-processor (SAES) for details on procedure to download the boot hardware key in the SAES.

### 50.3.8 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in Table 512: TAMP interconnection. The device secrets list is configurable thanks to TAMP resources protection configuration register (TAMP_RPCFGR).
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in Table 512: TAMP interconnection. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

### 50.3.9 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

*Note:* The backup registers are also erased when the product state is changed from Closed to Open.

**Tamper detection – confirmed mode**

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed
mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the Table 512: TAMP interconnection.

This mode is selected to erase automatically the device secrets when the tamper is detected.

**Tamper detection – potential mode**

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the Table 512: TAMP interconnection.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

**Potential tamper to confirmed tamper timeout**

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to Table 512: TAMP interconnection). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

**Device resources protection configuration**

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in Table 512: TAMP interconnection
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in Table 512: TAMP interconnection
Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set. It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

50.3.10 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00 and active mode is not selected). Refer to Section : Level detection with filtering on tamper inputs (passive mode).

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCCLK is stopped (it is set when RTCCCLK restarts).

Table 513. Device resource x tamper protection

<table>
<thead>
<tr>
<th>RPCFGx = 0</th>
<th>Potential tamper or BKBLOCK</th>
<th>Confirmed tamper or BKERASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>No effect on device resource x</td>
<td>No effect on device resource x</td>
<td></td>
</tr>
<tr>
<td>Device secret x protected as described by tamp_potential_rpcfgx(1)</td>
<td>Device secret x protected as described by tamp_confirmed_rpcfgx(1)</td>
<td></td>
</tr>
</tbody>
</table>

1. Refer to Table 512: TAMP interconnection.
Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when VDD power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection.

Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2ATPER × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.
PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods. The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

<table>
<thead>
<tr>
<th>Number of selected outputs</th>
<th>Minimum ATPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3 or 4</td>
<td>2</td>
</tr>
<tr>
<td>5 or more</td>
<td>3</td>
</tr>
</tbody>
</table>

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

As illustrated in Figure 679, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.
Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in Table 516 for each possible setting of ATCKSEL, assuming $f_{RTCCLK} = 32.768$ kHz.

### Table 516. Active tamper filtered pulse duration

<table>
<thead>
<tr>
<th>ATCKSEL[3:0]</th>
<th>CK_ATPRE frequency</th>
<th>Minimum filtered pulse width (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>$f_{RTCCLK}$</td>
<td>0.030</td>
</tr>
<tr>
<td>0x1</td>
<td>$f_{RTCCLK}/2$</td>
<td>0.061</td>
</tr>
<tr>
<td>0x2</td>
<td>$f_{RTCCLK}/4$</td>
<td>0.122</td>
</tr>
<tr>
<td>0x3</td>
<td>$f_{RTCCLK}/8$</td>
<td>0.244</td>
</tr>
<tr>
<td>0x4</td>
<td>$f_{RTCCLK}/16$</td>
<td>0.488</td>
</tr>
<tr>
<td>0x5</td>
<td>$f_{RTCCLK}/32$</td>
<td>0.977</td>
</tr>
<tr>
<td>0x6</td>
<td>$f_{RTCCLK}/64$</td>
<td>1.953</td>
</tr>
<tr>
<td>0x7</td>
<td>$f_{RTCCLK}/128$</td>
<td>3.906</td>
</tr>
<tr>
<td>0x8</td>
<td>$f_{RTCCLK}/2048$</td>
<td>62.500 (1)</td>
</tr>
</tbody>
</table>

1. This setting requires that \((PREDIV_A+1) = 128\) and \((PREDIV_S+1)\) is a multiple of 16.

**Note:** Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

**Caution:** Entering RTC initialization mode stops CK_ATPRE and CK_ATPER clocks when ATCKSEL[3] = 1. Therefore, TAMP_OUTy pin stops toggling until INIT mode exit.

Refer to section **Section : Calendar initialization and configuration**.

Refer also to **RTC alarm A subsecond register (RTC_ALRMASSR)**, **RTC alarm B subsecond register (RTC_ALRMBSSR)**, **RTC alarm A binary mode register (RTC_ALRABINR)** and **RTC alarm B binary mode register (RTC_ALRBBINR)** in case RTC binary mode is used in conjunction with ATCKSEL[3] = 1.

**Caution:** Caution: The active tamper detection is no more functional in case of calendar overflow when ATCKSEL[3] = 1. It is mandatory to enable the internal tamper 5 on calendar overflow to ensure tamper protection.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

### Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

### Table 516: Active tamper filtered pulse duration
Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

- If INITS = 0x1 (initialization already done):
  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.

- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

### 50.4 TAMP low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. TAMP interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop</td>
<td>No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Functional in all low-power modes</th>
<th>Functional in VBAT mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP_IN[8:1]</td>
<td>Yes</td>
<td>Yes/no(1)</td>
</tr>
<tr>
<td>TAMP_OUT[8:1]</td>
<td>Yes</td>
<td>Yes/no(2)</td>
</tr>
</tbody>
</table>

1. Only TAMP_IN1, TAMP_IN2 and TAMP_IN3 are functional in VBAT mode.
2. Only TAMP_OUT2 is functional in VBAT mode.
50.5 TAMP interrupts

Table 519. Interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag(1)</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from low-power modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAMP</td>
<td>Tamper x(2)</td>
<td>TAMPxF</td>
<td>TAMPxIE</td>
<td>Write 1 in CTAMPxF</td>
<td>Yes(3)</td>
</tr>
<tr>
<td></td>
<td>Internal tamper y(2)</td>
<td>ITAMPyF</td>
<td>ITAMPyIE</td>
<td>Write 1 in CITAMPyF</td>
<td>Yes(3)</td>
</tr>
</tbody>
</table>

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to Table 517: Effect of low-power modes on TAMP for more details about available features in the low-power modes.

50.6 TAMP registers

Refer to Section 1.2 of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

50.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x00
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>ITAMP15E: Internal tamper 15 enable</td>
</tr>
<tr>
<td>29</td>
<td>0: Internal tamper 15 disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 15 enabled.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 26 **ITAMP11E**: Internal tamper 11 enable
   0: Internal tamper 11 disabled.
   1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
   0: Internal tamper 9 disabled.
   1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
   0: Internal tamper 8 disabled.
   1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
   0: Internal tamper 7 disabled.
   1: Internal tamper 7 enabled.

Bit 21 **ITAMP6E**: Internal tamper 6 enable
   0: Internal tamper 6 disabled.
   1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
   0: Internal tamper 5 disabled.
   1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
   0: Internal tamper 4 disabled.
   1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
   0: Internal tamper 3 disabled.
   1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
   0: Internal tamper 2 disabled.
   1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
   0: Internal tamper 1 disabled.
   1: Internal tamper 1 enabled.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8E**: Tamper detection on TAMP.IN8 enable\(^{(1)}\)
   0: Tamper detection on TAMP.IN8 is disabled.
   1: Tamper detection on TAMP.IN8 is enabled.

Bit 6 **TAMP7E**: Tamper detection on TAMP.IN7 enable\(^{(1)}\)
   0: Tamper detection on TAMP.IN7 is disabled.
   1: Tamper detection on TAMP.IN7 is enabled.

Bit 5 **TAMP6E**: Tamper detection on TAMP.IN6 enable\(^{(1)}\)
   0: Tamper detection on TAMP.IN6 is disabled.
   1: Tamper detection on TAMP.IN6 is enabled.

Bit 4 **TAMP5E**: Tamper detection on TAMP.IN5 enable\(^{(1)}\)
   0: Tamper detection on TAMP.IN5 is disabled.
   1: Tamper detection on TAMP.IN5 is enabled.
### TAMP control register 2 (TAMP_CR2)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

| Bit 3 | TAMPA4TRG: Tamper detection on TAMP_IN4 enable
| Bit 2 | TAMPA3TRG: Tamper detection on TAMP_IN3 enable
| Bit 1 | TAMPA2TRG: Tamper detection on TAMP_IN2 enable
| Bit 0 | TAMPA1TRG: Tamper detection on TAMP_IN1 enable

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

| Bit 31 | TAMPA8TRG: Active level for tamper 8 input (active mode disabled)
| Bit 30 | TAMPA7TRG: Active level for tamper 7 input (active mode disabled)
| Bit 29 | TAMPA6TRG: Active level for tamper 6 input (active mode disabled)

---

| Bit 31 | TAMPA8TRG: Active level for tamper 8 input (active mode disabled)
| Bit 30 | TAMPA7TRG: Active level for tamper 7 input (active mode disabled)
| Bit 29 | TAMPA6TRG: Active level for tamper 6 input (active mode disabled)

---

Bit 31 **TAMPA8TRG**: Active level for tamper 8 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 tamper 8 input staying low triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 8 input rising high triggers a tamper detection event.

Bit 30 **TAMPA7TRG**: Active level for tamper 7 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 tamper 7 input staying low triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 7 input rising high triggers a tamper detection event.

Bit 29 **TAMPA6TRG**: Active level for tamper 6 input (active mode disabled)

- 0: If TAMPFLT ≠ 00 tamper 6 input staying low triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 6 input rising high triggers a tamper detection event.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)
  0: If TAMPFLT ≠ 00 tamper 5 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.
  1: If TAMPFLT ≠ 00 tamper 5 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 5 input falling edge triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)
  0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.
  1: If TAMPFLT ≠ 00 tamper 4 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input
  0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.
  1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input
  0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.
  1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input
  0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.
  1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase
  Writing “1” to this bit reset the backup registers and device secrets(1). Writing 0 has no effect.
  This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked
  0: backup registers and device secrets(1) can be accessed if no tamper flag is set
  1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask
  0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
  1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
  *The tamper 3 interrupt must not be enabled when TAMP3MSK is set.*

Bit 17 **TAMP2MSK**: Tamper 2 mask
  0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
  1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
  *The tamper 2 interrupt must not be enabled when TAMP2MSK is set.*
Bit 16 **TAMP1MSK**: Tamper 1 mask

0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.

1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets\(^1\) are not erased.

*The tamper 1 interrupt must not be enabled when TAMP1MSK is set.*

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8POM**: Tamper 8 potential mode

0: Tamper 8 event detection is in confirmed mode\(^1\).

1: Tamper 8 event detection is in potential mode\(^2\).

Bit 6 **TAMP7POM**: Tamper 7 potential mode

0: Tamper 7 event detection is in confirmed mode\(^1\).

1: Tamper 7 event detection is in potential mode\(^2\).

Bit 5 **TAMP6POM**: Tamper 6 potential mode

0: Tamper 6 event detection is in confirmed mode\(^1\).

1: Tamper 6 event detection is in potential mode\(^2\).

Bit 4 **TAMP5POM**: Tamper 5 potential mode

0: Tamper 5 event detection is in confirmed mode\(^1\).

1: Tamper 5 event detection is in potential mode\(^2\).

Bit 3 **TAMP4POM**: Tamper 4 potential mode

0: Tamper 4 event detection is in confirmed mode\(^1\).

1: Tamper 4 event detection is in potential mode\(^2\).

Bit 2 **TAMP3POM**: Tamper 3 potential mode

0: Tamper 3 event detection is in confirmed mode\(^1\).

1: Tamper 3 event detection is in potential mode\(^2\).

Bit 1 **TAMP2POM**: Tamper 2 potential mode

0: Tamper 2 event detection is in confirmed mode\(^1\).

1: Tamper 2 event detection is in potential mode\(^2\).

Bit 0 **TAMP1POM**: Tamper 1 potential mode

0: Tamper 1 event detection is in confirmed mode\(^1\).

1: Tamper 1 event detection is in potential mode\(^2\).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 512: TAMP interconnection.

2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 512: TAMP interconnection.

### 50.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against non-privileged access. Refer to Section 50.3.6: *TAMP privilege protection modes.*
Tamper and backup registers (TAMP)

Address offset: 0x08
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP15POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP11POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP9POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP7POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP5POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP3POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP2POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ITAMP1POM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  Reserved, must be kept at reset value.

Bit 14  **ITAMP15POM**: Internal tamper 15 potential mode
0: Internal tamper 15 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 15 event detection is in potential mode\(^{(2)}\).

Bit 13  Reserved, must be kept at reset value.

Bit 12  Reserved, must be kept at reset value.

Bit 11  Reserved, must be kept at reset value.

Bit 10  **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 11 event detection is in potential mode\(^{(2)}\).

Bit 9  Reserved, must be kept at reset value.

Bit 8  **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 9 event detection is in potential mode\(^{(2)}\).

Bit 7  **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 8 event detection is in potential mode\(^{(2)}\).

Bit 6  **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 7 event detection is in potential mode\(^{(2)}\).

Bit 5  **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 6 event detection is in potential mode\(^{(2)}\).

Bit 4  **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 5 event detection is in potential mode\(^{(2)}\).

Bit 3  **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 4 event detection is in potential mode\(^{(2)}\).
Bit 2 ITAMP3POM: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 3 event detection is in potential mode\(^{(2)}\).

Bit 1 ITAMP2POM: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 2 event detection is in potential mode\(^{(2)}\).

Bit 0 ITAMP1POM: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode\(^{(1)}\).
1: Internal tamper 1 event detection is in potential mode\(^{(2)}\).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in Table 512: TAMP interconnection.
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in Table 512: TAMP interconnection.

50.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0xOC
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

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

Bit 7 TAMPPUDIS: TAMP_INx pull-up disable
This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP_INx pins.
Bits 6:5 TAMPPRCH[1:0]: TAMP_INx precharge duration
These bits determine the duration of time during which the pull-up is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 4:3 TAMPFILT[1:0]: TAMP_INx filter count
These bits determine the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFILT is valid for each of the TAMP_INx inputs.
0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 TAMPFREQ[2:0]: Tamper sampling frequency
Determines the frequency at which each of the TAMP_INx inputs are sampled.
0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

50.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x10
Backup domain reset value: 0x0007 0000
System reset: not affected

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>ATOSEL4[1:0]</td>
<td>ATOSEL3[1:0]</td>
<td>ATOSEL2[1:0]</td>
<td>ATOSEL1[1:0]</td>
<td>TAMP8 AM</td>
<td>TAMP7 AM</td>
<td>TAMP6 AM</td>
<td>TAMP5 AM</td>
<td>TAMP4 AM</td>
<td>TAMP3 AM</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31  **FLTEN**: Active tamper filter enable
0: Active tamper filtering disable
1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30  **ATOSHARE**: Active tamper output sharing
0: Each active tamper input TAMP\_IN\_i is compared with its dedicated output TAMP\_OUT\_i
1: Each active tamper input TAMP\_IN\_i is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27  Reserved, must be kept at reset value.

Bits 26:24  **ATPER[2:0]**: Active tamper output change period
The tamper output is changed every CK\_ATPER = (2\(^{ATPER} \times CK\_ATPRE\)) cycles. Refer to Table 515: **Minimum ATPER value**.

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

Bits 19:16  **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection
These bits select the RTC asynchronous prescaler stage output. The selected clock is CK\_ATPRE.
0000: RTCCLOCK is selected
0001: RTCCLOCK/2 is selected
0010: RTCCLOCK/4 is selected
0011: RTCCLOCK/8 is selected
0100: RTCCLOCK/16 is selected
0101: RTCCLOCK/32 is selected
0110: RTCCLOCK/64 is selected
0111: RTCCLOCK/128 is selected
1011: RTCCLOCK/2048 is selected when (PREDIV\_A+1) = 128 and (PREDIV\_S+1) is a multiple of 16.
Others: Reserved

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK\_ATPRE cycles after all the active tampers are disable.

Bits 15:14  **ATOSEL4[1:0]**: Active tamper shared output 4 selection
00: TAMPOUTSELi4 = TAMP\_OUT1
01: TAMPOUTSELi4 = TAMP\_OUT2
10: TAMPOUTSELi4 = TAMP\_OUT3
11: TAMPOUTSELi4 = TAMP\_OUT4
If the TAMP\_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 13:12  **ATOSEL3[1:0]**: Active tamper shared output 3 selection
00: TAMPOUTSELi3 = TAMP\_OUT1
01: TAMPOUTSELi3 = TAMP\_OUT2
10: TAMPOUTSELi3 = TAMP\_OUT3
11: TAMPOUTSELi3 = TAMP\_OUT4
If the TAMP\_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.
**TAMPER and backup registers (TAMP)**

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection

- 00: TAMPOUTSEL2 = TAMP_OUT1
- 01: TAMPOUTSEL2 = TAMP_OUT2
- 10: TAMPOUTSEL2 = TAMP_OUT3
- 11: TAMPOUTSEL2 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection

- 00: TAMPOUTSEL1 = TAMP_OUT1
- 01: TAMPOUTSEL1 = TAMP_OUT2
- 10: TAMPOUTSEL1 = TAMP_OUT3
- 11: TAMPOUTSEL1 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bit 7 **TAMP8AM**: Tamper 8 active mode

- 0: Tamper 8 detection mode is passive.
- 1: Tamper 8 detection mode is active.

Bit 6 **TAMP7AM**: Tamper 7 active mode

- 0: Tamper 7 detection mode is passive.
- 1: Tamper 7 detection mode is active.

Bit 5 **TAMP6AM**: Tamper 6 active mode

- 0: Tamper 6 detection mode is passive.
- 1: Tamper 6 detection mode is active.

Bit 4 **TAMP5AM**: Tamper 5 active mode

- 0: Tamper 5 detection mode is passive.
- 1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode

- 0: Tamper 4 detection mode is passive.
- 1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode

- 0: Tamper 3 detection mode is passive.
- 1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode

- 0: Tamper 2 detection mode is passive.
- 1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode

- 0: Tamper 1 detection mode is passive.
- 1: Tamper 1 detection mode is active.

**Note:** Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.
50.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 SEED[31:0]: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

50.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>


| r  | r  | r  | r  | r  | r  | r  | r  |

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 INITS: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.
Bit 14  **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8  Reserved, must be kept at reset value.

Bits 7:0  **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

### 50.6.8  TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against non-privileged access. Refer to  *Section 50.3.6: TAMP privilege protection modes*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ATOSSEL3[1:0]</td>
<td>ATOSSEL2[2:0]</td>
<td>ATOSSEL1[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:29  **ATOSEL8[2:0]**: Active tamper shared output 8 selection

- 000: TAMPOUTSEL8 = TAMP_OUT1
- 001: TAMPOUTSEL8 = TAMP_OUT2
- 010: TAMPOUTSEL8 = TAMP_OUT3
- 011: TAMPOUTSEL8 = TAMP_OUT4
- 100: TAMPOUTSEL8 = TAMP_OUT5
- 101: TAMPOUTSEL8 = TAMP_OUT6
- 110: TAMPOUTSEL8 = TAMP_OUT7
- 111: TAMPOUTSEL8 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 28:26  **ATOSEL7[2:0]**: Active tamper shared output 7 selection

- 000: TAMPOUTSEL7 = TAMP_OUT1
- 001: TAMPOUTSEL7 = TAMP_OUT2
- 010: TAMPOUTSEL7 = TAMP_OUT3
- 011: TAMPOUTSEL7 = TAMP_OUT4
- 100: TAMPOUTSEL7 = TAMP_OUT5
- 101: TAMPOUTSEL7 = TAMP_OUT6
- 110: TAMPOUTSEL7 = TAMP_OUT7
- 111: TAMPOUTSEL7 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 25:23  **ATOSEL6[2:0]**: Active tamper shared output 6 selection

- 000: TAMPOUTSEL6 = TAMP_OUT1
- 001: TAMPOUTSEL6 = TAMP_OUT2
- 010: TAMPOUTSEL6 = TAMP_OUT3
- 011: TAMPOUTSEL6 = TAMP_OUT4
- 100: TAMPOUTSEL6 = TAMP_OUT5
- 101: TAMPOUTSEL6 = TAMP_OUT6
- 110: TAMPOUTSEL6 = TAMP_OUT7
- 111: TAMPOUTSEL6 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 22:20  **ATOSEL5[2:0]**: Active tamper shared output 5 selection

- 000: TAMPOUTSEL5 = TAMP_OUT1
- 001: TAMPOUTSEL5 = TAMP_OUT2
- 010: TAMPOUTSEL5 = TAMP_OUT3
- 011: TAMPOUTSEL5 = TAMP_OUT4
- 100: TAMPOUTSEL5 = TAMP_OUT5
- 101: TAMPOUTSEL5 = TAMP_OUT6
- 110: TAMPOUTSEL5 = TAMP_OUT7
- 111: TAMPOUTSEL5 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.
Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection
- 000: TAMPOUTSEL4 = TAMP_OUT1
- 001: TAMPOUTSEL4 = TAMP_OUT2
- 010: TAMPOUTSEL4 = TAMP_OUT3
- 011: TAMPOUTSEL4 = TAMP_OUT4
- 100: TAMPOUTSEL4 = TAMP_OUT5
- 101: TAMPOUTSEL4 = TAMP_OUT6
- 110: TAMPOUTSEL4 = TAMP_OUT7
- 111: TAMPOUTSEL4 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 18:17 are the mirror of ATOSEL4[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection
- 000: TAMPOUTSEL3 = TAMP_OUT1
- 001: TAMPOUTSEL3 = TAMP_OUT2
- 010: TAMPOUTSEL3 = TAMP_OUT3
- 011: TAMPOUTSEL3 = TAMP_OUT4
- 100: TAMPOUTSEL3 = TAMP_OUT5
- 101: TAMPOUTSEL3 = TAMP_OUT6
- 110: TAMPOUTSEL3 = TAMP_OUT7
- 111: TAMPOUTSEL3 = TAMP_OUT8

If the TAMP_OUTx output is not available in the package pinout, the output selection value is reserved and must not be used.

Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.
Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

50.6.9 TAMP configuration register (TAMP_CFGCR)

This register can be globally write-protected, or each bit of this register can be individually write-protected against non-privileged access depending on the TAMP_PRIVCFGFR configuration (refer to Section 50.3.6: TAMP privilege protection modes).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected
Bit 31  Reserved, must be kept at reset value.

Bit 30  BHKLOCK: Boot hardware key lock
This bit can be read and can only be written to 1 by software. It is cleared by hardware together with the backup registers following a tamper detection event or when the product state is opened.
0: The Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the Protection zone they belong to.
1: The backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read nor in write (they are read as 0 and write ignore).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16  BKPW[7:0]: Backup registers write protection offset
BKPW value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPW) to TAMP_BKPzR (z = BKPW-1, with BKPW > BKPW).
If BKPWSEC = 0 or if BKPWSEC ≤ BKPRWSSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPyR (t = BKPW if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).
If BKPWSEC = 32: there is no protection zone 3.

Refer to Figure 678: Backup registers protection zones.

Note: If BKPWPRIV is set, BKPW[7:0] can be written only in privileged mode.

Bit 15  Reserved, must be kept at reset value.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0  BKPRW[7:0]: Backup registers read/write protection offset
BKPRW value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRW-1, with BKPRW ≥ 1).
If BKPRW = 0: there is no protection zone 1.

Refer to Figure 678: Backup registers protection zones.

Note: If BKPRWPRIV is set, BKPRW[7:0] can be written only in privileged mode.

50.6.10  TAMPPRIV Privilege configuration register (TAMP_PRIVCFG)
This register can be written only when the APB access is privileged.
Address offset: 0x24
Backup domain reset value: 0x0000 0000
System reset: not affected

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding backup registers)
- 0: Tamper configuration and interrupt can be written with privileged or unprivileged access.
- 1: Tamper configuration and interrupt can be written only with privileged access.

*Note: Refer to Section 50.3.6: TAMP privilege protection modes for details on the read protection.*

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection
- 0: Backup registers zone 2 can be written with privileged or unprivileged access.
- 1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection
- 0: Backup registers zone 1 can be read and written with privileged or unprivileged access.
- 1: Backup registers zone 1 can be read and written only with privileged access.

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection
- 0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or non-privileged.
- 1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

### 50.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against non-privileged access. Refer to *Section 50.3.6:*
**Tamper and backup registers (TAMP)**

**TAMP privilege protection modes.**

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 30</td>
<td><strong>ITAMP15IE</strong>: Internal tamper 15 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 15 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 15 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 26</td>
<td><strong>ITAMP11IE</strong>: Internal tamper 11 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 11 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 11 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 24</td>
<td><strong>ITAMP9IE</strong>: Internal tamper 9 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 9 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 9 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 23</td>
<td><strong>ITAMP8IE</strong>: Internal tamper 8 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 8 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 8 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 22</td>
<td><strong>ITAMP7IE</strong>: Internal tamper 7 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 7 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 7 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 21</td>
<td><strong>ITAMP6IE</strong>: Internal tamper 6 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 6 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 6 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>ITAMP5IE</strong>: Internal tamper 5 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 5 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 5 interrupt enabled.</td>
</tr>
<tr>
<td>Bit 19</td>
<td><strong>ITAMP4IE</strong>: Internal tamper 4 interrupt enable</td>
</tr>
<tr>
<td></td>
<td>0: Internal tamper 4 interrupt disabled.</td>
</tr>
<tr>
<td></td>
<td>1: Internal tamper 4 interrupt enabled.</td>
</tr>
</tbody>
</table>
Bit 18  **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17  **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16  **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled.

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **TAMP8IE**: Tamper 8 interrupt enable
0: Tamper 8 interrupt disabled.
1: Tamper 8 interrupt enabled.

Bit 6  **TAMP7IE**: Tamper 7 interrupt enable
0: Tamper 7 interrupt disabled.
1: Tamper 7 interrupt enabled.

Bit 5  **TAMP6IE**: Tamper 6 interrupt enable
0: Tamper 6 interrupt disabled.
1: Tamper 6 interrupt enabled.

Bit 4  **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3  **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2  **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled.

Bit 1  **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0  **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

### 50.6.12  **TAMP status register (TAMP_SR)**

This register can be protected against non-privileged access. Refer to *Section 50.3.6:*
**TAMP privilege protection modes.**

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bit 30 **ITAMP15F**: Internal tamper 15 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 15.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11F**: Internal tamper 11 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.
Bit 18 **ITAMP3F**: Internal tamper 3 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **TAMP8F**: TAMP8 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP8 input.

Bit 6 **TAMP7F**: TAMP7 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP7 input.

Bit 5 **TAMP6F**: TAMP6 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP6 input.

Bit 4 **TAMP5F**: TAMP5 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

### 50.6.13 TAMPER masked interrupt status register (TAMP_MISR)

This register can be protected against non-privileged access. Refer to *Section 50.3.6: TAMPER privilege protection modes.*

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected
Bit 31  Reserved, must be kept at reset value.

Bit 30  **ITAMP15MF**: internal tamper 15 interrupt masked flag  
This flag is set by hardware when the internal tamper 15 interrupt is raised.

Bit 29  Reserved, must be kept at reset value.

Bit 28  Reserved, must be kept at reset value.

Bit 27  Reserved, must be kept at reset value.

Bit 26  **ITAMP11MF**: internal tamper 11 interrupt masked flag  
This flag is set by hardware when the internal tamper 11 interrupt is raised.

Bit 25  Reserved, must be kept at reset value.

Bit 24  **ITAMP9MF**: internal tamper 9 interrupt masked flag  
This flag is set by hardware when the internal tamper 9 interrupt is raised.

Bit 23  **ITAMP8MF**: Internal tamper 8 interrupt masked flag  
This flag is set by hardware when the internal tamper 8 interrupt is raised.

Bit 22  **ITAMP7MF**: Internal tamper 7 tamper interrupt masked flag  
This flag is set by hardware when the internal tamper 7 interrupt is raised.

Bit 21  **ITAMP6MF**: Internal tamper 6 interrupt masked flag  
This flag is set by hardware when the internal tamper 6 interrupt is raised.

Bit 20  **ITAMP5MF**: Internal tamper 5 interrupt masked flag  
This flag is set by hardware when the internal tamper 5 interrupt is raised.

Bit 19  **ITAMP4MF**: Internal tamper 4 interrupt masked flag  
This flag is set by hardware when the internal tamper 4 interrupt is raised.

Bit 18  **ITAMP3MF**: Internal tamper 3 interrupt masked flag  
This flag is set by hardware when the internal tamper 3 interrupt is raised.

Bit 17  **ITAMP2MF**: Internal tamper 2 interrupt masked flag  
This flag is set by hardware when the internal tamper 2 interrupt is raised.

Bit 16  **ITAMP1MF**: Internal tamper 1 interrupt masked flag  
This flag is set by hardware when the internal tamper 1 interrupt is raised.

Bits 15:8  Reserved, must be kept at reset value.

Bit 7  **TAMP8MF**: TAMP8 interrupt masked flag  
This flag is set by hardware when the tamper 8 interrupt is raised.

Bit 6  **TAMP7MF**: TAMP7 interrupt masked flag  
This flag is set by hardware when the tamper 7 interrupt is raised.

Bit 5  **TAMP6MF**: TAMP6 interrupt masked flag  
This flag is set by hardware when the tamper 6 interrupt is raised.

Bit 4  **TAMP5MF**: TAMP5 interrupt masked flag  
This flag is set by hardware when the tamper 5 interrupt is raised.

Bit 3  **TAMP4MF**: TAMP4 interrupt masked flag  
This flag is set by hardware when the tamper 4 interrupt is raised.
### TAMP status clear register (TAMP_SCR)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x3C

System reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>30</td>
<td>CITAMP15F: Clear ITAMP15 detection flag</td>
<td>w</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>26</td>
<td>CITAMP11F: Clear ITAMP11 detection flag</td>
<td>w</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>CITAMP9F: Clear ITAMP9 detection flag</td>
<td>w</td>
</tr>
<tr>
<td>23</td>
<td>CITAMP8F: Clear ITAMP8 detection flag</td>
<td>w</td>
</tr>
<tr>
<td>22</td>
<td>CITAMP7F: Clear ITAMP7 detection flag</td>
<td>w</td>
</tr>
<tr>
<td>21</td>
<td>CITAMP6F: Clear ITAMP6 detection flag</td>
<td>w</td>
</tr>
<tr>
<td>20</td>
<td>CITAMP5F: Clear ITAMP5 detection flag</td>
<td>w</td>
</tr>
</tbody>
</table>
Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
  Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
  Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
  Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
  Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CTAMP8F**: Clear TAMP8 detection flag
  Writing 1 in this bit clears the TAMP8F bit in the TAMP_SR register.

Bit 6 **CTAMP7F**: Clear TAMP7 detection flag
  Writing 1 in this bit clears the TAMP7F bit in the TAMP_SR register.

Bit 5 **CTAMP6F**: Clear TAMP6 detection flag
  Writing 1 in this bit clears the TAMP6F bit in the TAMP_SR register.

Bit 4 **CTAMP5F**: Clear TAMP5 detection flag
  Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
  Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
  Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
  Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
  Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.
50.6.15 **TAMP monotonic counter 1 register (TAMP_COUNT1R)**

This register can be protected against non-privileged access. Refer to *Section 50.3.6: TAMP privilege protection modes.*

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>COUNT[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
<tr>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COUNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **COUNT[31:0]:**

This register is read-only only and is incremented by one when a write access is done to this register. This register cannot roll-over and is frozen when reaching the maximum value.

50.6.16 **TAMP resources protection configuration register (TAMP_RPCFGR)**

This register can be protected against non-privileged access. Refer to *Section 50.3.6: TAMP privilege protection modes.*

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

<table>
<thead>
<tr>
<th>Bit 31 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 30:8 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 7 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 6 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 5 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4 Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 3 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 2  Reserved, must be kept at reset value.
Bit 1  Reserved, must be kept at reset value.
Bit 0  **RPCFG0**: Configurable resource 0 protection
        0: Resource 0 is not included in the device secrets protected by TAMP peripheral
        1: Resource 0 is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_confirmed_rpcfg0 and tamp_potential_rpcfg0 signals in Table 510: TAMP input/output pins and Table 512: TAMP interconnection.

### 50.6.17 TAMP backup x register (TAMP_BKPxR)

This register can be protected against non-privileged access. Refer to Section 50.3.6: TAMP privilege protection modes.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)
Backup domain reset value: 0x0000 0000
System reset: not affected

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>BKP[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>BKP[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **BKP[31:0]**:
The application can write or read data to and from these registers.
In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set. This register is also reset when the product state is opened.
### Table 520. TAMP register map and reset values

| Offset | Register name    | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | TAMP_CR1         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x04   | TAMP_CR2         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x08   | TAMP_CR3         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0C   | TAMP_FLTCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10   | TAMP_ATCR1       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x14   | TAMP_ATSEEDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x18   | TAMP_ATOR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1C   | TAMP_ATCR2       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |    |
| 0x20   | TAMP_CFGR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x24   | TAMP_PRIVCFGFR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value      | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 for the register boundary addresses.
51  Inter-integrated circuit interface (I2C)

51.1  Introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-
integrated circuit) bus. It provides multimaster capability, and controls all I²C-bus-specific
sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode
(Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power
management bus) compatible.

It can use DMA to reduce the CPU load.

51.2  I2C main features

- I²C-bus specification rev03 compatibility:
  - Slave and master modes
  - Multimaster capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit slave addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility:
  - Hardware PEC (packet error checking) generation and verification with ACK
    control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock
- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to Section 51.3: I2C implementation.
51.3 **I2C implementation**

This section provides an implementation overview with respect to the I2C instantiation.

### Table 521. I2C implementation

<table>
<thead>
<tr>
<th>I2C features(1)</th>
<th>I2C1</th>
<th>I2C2</th>
<th>I2C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10-bit addressing mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Standard-mode (up to 100 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode (up to 400 kbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Independent clock</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wake-up from Stop mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SMBus/PMBus</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

1. X = supported.

51.4 **I2C functional description**

In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.
51.4.1 I2C block diagram

Figure 680. Block diagram

51.4.2 I2C pins and internal signals

Table 522. I2C input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_SDA</td>
<td>Bidirectional</td>
<td>I²C-bus data</td>
</tr>
<tr>
<td>I2C_SCL</td>
<td>Bidirectional</td>
<td>I²C-bus clock</td>
</tr>
<tr>
<td>I2C_SMBA</td>
<td>Bidirectional</td>
<td>SMBus alert</td>
</tr>
</tbody>
</table>
51.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period \( t_{I2CCLK} \) must respect the following conditions:

\[
\begin{align*}
    t_{I2CCLK} &< (t_{LOW} - t_{filters}) / 4 \\
    t_{I2CCLK} &< t_{HIGH}
\end{align*}
\]

where \( t_{LOW} \) is the SCL low time, \( t_{HIGH} \) is the SCL high time, and \( t_{filters} \) is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x \( t_{I2CCLK} \).

The i2c_pclk clock period \( t_{PCLK} \) must respect the condition \( t_{PCLK} < 4/3 t_{SCL} \), where \( t_{SCL} \) is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for \( t_{I2CCLK} \).

51.4.4 Mode selection

The peripheral can operate as:

- slave transmitter
- slave receiver
- master transmitter
- master receiver

By default, the peripheral operates in slave mode. It automatically switches from slave to master mode upon generating START condition, and from master to slave mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multimaster I²C-bus environment.

Communication flow

In master mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in master mode by software.

In slave mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.
Data and addresses are transferred as 8-bit bytes, MSB first. The first one (7-bit addressing) or two (10-bit addressing) bytes following the START condition contain the address. The address is always transmitted in master mode.

The following figure shows the transmission of a single byte. The master generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

![Figure 681. I²C-bus protocol](Image)

The acknowledge can be enabled or disabled by software. The own addresses of the I²C peripheral can be selected by software.

### 51.4.5 I²C initialization

#### Enabling and disabling the peripheral

Before enabling the I²C peripheral, configure and enable its clock through the clock controller, and initialize its control registers.

The I²C peripheral can then be enabled by setting the PE bit of the I²C_CR1 register.

Disabling the I²C peripheral by clearing the PE bit resets the I²C peripheral. Refer to **Section 51.4.6** for more details.

#### Noise filters

Before enabling the I²C peripheral by setting the PE bit of the I²C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I²C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

<table>
<thead>
<tr>
<th></th>
<th>SDA</th>
<th>SCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start condition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop condition</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Caution:  The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C configuration window.

<table>
<thead>
<tr>
<th>Item</th>
<th>Analog filter</th>
<th>Digital filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filtering capacity(1)</td>
<td>≥ 50 ns</td>
<td>One to fifteen i2c_ker_ck periods</td>
</tr>
<tr>
<td>Benefits</td>
<td>Available in Stop mode</td>
<td>– Programmable filtering capacity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Extra filtering capability versus I²C-bus specification requirements</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Stable filtering capacity</td>
</tr>
<tr>
<td>Drawbacks</td>
<td>Filtering capacity variation with temperature, voltage, and silicon process</td>
<td>Wake-up from Stop mode on address match not supported when the digital filter is enabled</td>
</tr>
</tbody>
</table>

1. Maximum duration of spikes that the filter can suppress
When the SCL falling edge is internally detected, the delay $t_{SDADEL}$ (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

$$t_{SDADEL} = SDADEL \times t_{PRESC} + t_{I2CCLK}, \text{ where } t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}.$$ 

The total SDA output delay is:

$$t_{SYNC1} + \{(SDADEL \times (PRESC + 1) + 1) \times t_{I2CCLK}\}$$

The $t_{SYNC1}$ duration depends upon:

- SCL falling slope
- input delay $t_{AF(\text{min})} < t_{AF} < t_{AF(\text{max})}$ introduced by the analog filter (if enabled)
- input delay $t_{DNF} = DNF \times t_{I2CCLK}$ introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$$\frac{t_{I}(\text{max}) + t_{HD;DAT(\text{min})} - t_{AF(\text{min})} - \{(DNF + 3) \times t_{I2CCLK}\}}{(PRESC + 1) \times t_{I2CCLK}} \leq SDADEL$$

$$SDADEL \leq \frac{t_{I}(HD;DAT(\text{max})} - t_{AF(\text{max})} - \{(DNF + 4) \times t_{I2CCLK}\}}{(PRESC + 1) \times t_{I2CCLK}}$$
Note: \( t_{AF\text{(min)}} \) and \( t_{AF\text{(max)}} \) are only part of the condition when the analog filter is enabled. Refer to the device datasheet for \( t_{AF} \) values.

The \( t_{HD;DAT} \) time can at maximum be 3.45 \( \mu \)s for Standard-mode, 0.9 \( \mu \)s for Fast-mode, and 0.45 \( \mu \)s for Fast-mode Plus. It must be lower than the maximum of \( t_{VD;DAT} \) by a transition time. This maximum must only be met if the device does not stretch the LOW period (\( t_{LOW} \)) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

\[
SDADEL \leq (t_{VD;DAT \text{ (max)}} - t_{r \text{ (max)}} - t_{AF \text{ (max)}} - [(DNF + 4) \times t_{I2CCLK}]) / ((PRESC + 1) \times t_{I2CCLK})
\]

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After \( t_{SDADEL} \), or after sending SDA output when the slave had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is \( t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC} \), where \( t_{PRESC} = (PRESC + 1) \times t_{I2CCLK} \). \( t_{SCLDEL} \) impacts the setup time \( t_{SU;DAT} \).

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

\[
(t_{r \text{ (max)}} + t_{SU;DAT \text{ (min)}}) / ((PRESC + 1) \times t_{I2CCLK}) \leq 1 - SCLDEL
\]

Refer to the following table for \( t_{r} \), \( t_{HD;DAT} \), \( t_{VD;DAT} \), \( t_{SU;DAT} \), and \( t_{SU;DAT} \) standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as master or slave stretches SCL low during at least \( [(SDADEL + SCLDEL + 1) \times (PRESC + 1) + 1] \times t_{I2CCLK} \) in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in slave mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

---

**Table 525. I²C-bus and SMBus specification data setup and hold times**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBus</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>( t_{HD;DAT} )</td>
<td>Data hold time</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
<tr>
<td>( t_{VD;DAT} )</td>
<td>Data valid time</td>
<td>-</td>
<td>3.45</td>
<td>-</td>
<td>0.9</td>
</tr>
<tr>
<td>( t_{SU;DAT} )</td>
<td>Data setup time</td>
<td>250</td>
<td>-</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>( t_{r} )</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
</tr>
<tr>
<td>( t_{r} )</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>300</td>
</tr>
</tbody>
</table>
Additionally, in master mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCCL[7:0] bitfields of the I2C_TIMINGR register.

When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay $t_{SCLL} = (SCLL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. The $t_{SCLL}$ delay impacts the SCL low time $t_{LOW}$.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay $t_{SCLH} = (SCLH + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. The $t_{SCLH}$ impacts the SCL high time $t_{HIGH}$.

Refer to [I2C master initialization](#) for more details.

**Caution:** Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the slave NOSTRETCH settings must also be done before enabling the peripheral. Refer to [I2C slave initialization](#) for more details.

---

**Figure 683. I2C initialization flow**

1. **Initial settings**
   - Clear PE bit in I2C_CR1

2. **Configure ANOFF and DNF[3:0] in I2C_CR1**

3. **Configure PRESC[3:0], SDADEL[3:0], SCLDEL[3:0], SCLH[7:0], and SCCL[7:0] in I2C_TIMINGR**

4. **Configure NOSTRETCH in I2C_CR1**

5. **Set PE bit in I2C_CR1**

   **End**

---

### 51.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR
PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:
1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

51.4.7 Data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and ninth SCL pulse (before the acknowledge pulse).

Figure 684. Data reception
Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 685. Data transmission

Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in master mode
- ACK control in slave receiver mode
- PEC generation/checking

In master mode, the byte counter is always used. By default, it is disabled in slave mode. It can be enabled by software, by setting the SBC (slave byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.
When RELOAD = 0 in master mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the master automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.

- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the master wants to send a RESTART condition.

**Caution:** The AUTOEND bit has no effect when the RELOAD bit is set.

### Table 526. I2C configuration

<table>
<thead>
<tr>
<th>Function</th>
<th>SBC bit</th>
<th>RELOAD bit</th>
<th>AUTOEND bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Tx/Rx NBYTES + STOP</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Master Tx/Rx + NBYTES + RESTART</td>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Slave Tx/Rx, all received bytes ACKed</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Slave Rx with ACK control</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

**51.4.8 I2C slave mode**

**I2C slave initialization**

To work in slave mode, the user must enable at least one slave address. The I2C_OAR1 and I2C_OAR2 registers are available to program the slave own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional slave addresses are required, the second slave address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the slave uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the master does not
support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

**Slave with clock stretching**

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus slave stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled slave addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In slave byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after \([(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] \times t_{I2CCLK}\) period.

**Slave without clock stretching**

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus slave does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.
**Slave byte control mode**

To allow byte ACK control in slave reception mode, the slave byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in slave reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and ninth SCL pulses. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent, and the next byte can be received.

NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

*Note:* The SBC bit must be configured when the I2C peripheral is disabled, when the slave is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

*Caution:* The slave byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

---

**Figure 686. Slave initialization flow**

1. SBC must be set to support SMBus features.

---

Initial settings

Clear OA1EN and OA2EN in I2C_OAR1/I2C_OAR2

Configure OA1[9:0], OA1MODE, OA1EN, OA2[6:0], OA2MSK[2:0], OA2EN, and GCEN

Optional:Configure SBC in I2C_CR1(1)

Enable interrupts and/or DMA in I2C_CR1

End
Slave transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The slave automatically releases the SCL and SDA lines to let the master perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the slave address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In slave byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.
Figure 687. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 0

1. **Slave transmission**
   - **Slave initialization**
2. Check if `I2C_ISR.ADDR = 1`?
   - **No**
   - **Yes**
3. Read ADDCODE and DIR in I2C_ISR
   - Optional: Set `I2C_ISR.TXE = 1`
   - Set `I2C_ICR.ADDRCF`
4. Check if `I2C_ISR.TXIS = 1`?
   - **No**
   - **Yes**
5. Write `I2C_TXDR.TXDATA`
Figure 688. Transfer sequence flow for I2C slave transmitter, NOSTRETCH = 1

Slave initialization

Slave transmission

I2C_ISR.TXIS = 1?

No

Yes

Write I2C_TXDR.TXDATA

I2C_ISR.TXIS = 1?

No

Yes

Set I2C_ICR.STOPCF

I2C_ISR.STOPF = 1?

No

Yes

Optional: Set I2C_ISR.TXE = 1 and I2C_ISR.TXIS=1
Figure 689. Transfer bus diagrams for I2C slave transmitter (mandatory events only)

Example I2C slave transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes without 1st data flush, NOSTRETCH=0:

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C slave transmitter 3 bytes, NOSTRETCH=1:

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF
Slave receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When a STOP is received and STOPIE is set in I2C_CR1, STOPF is set in I2C_ISR and an interrupt is generated.

Figure 690. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 0
**Figure 691. Transfer sequence flow for I2C slave receiver, NOSTRETCH = 1**

Slave reception → Slave initialization →

- **I2C_ISR.RXNE = 1?**
  - No → Read I2C_RXDR.RXDATA
  - Yes → I2C_ISR.STOPF = 1?
    - No → Yes
    - Yes → Set I2C_ICR.STOPCF

**Figure 692. Transfer bus diagrams for I2C slave receiver (mandatory events only)**

Example I2C slave receiver 3 bytes, NOSTRETCH = 0:

- **EV1**: ADDR ISR: check ADDCODE and DIR, set ADDRCF
- **EV2**: RXNE ISR: rd data1
- **EV3**: RXNE ISR: rd data2
- **EV4**: RXNE ISR: rd data3

Example I2C slave receiver 3 bytes, NOSTRETCH = 1:

- **EV1**: RXNE ISR: rd data1
- **EV2**: RXNE ISR: rd data2
- **EV3**: RXNE ISR: rd data3

**Legend**
- Transmission
- Reception
- SCL stretch
51.4.9 I2C master mode

I2C master initialization

Before enabling the peripheral, the I2C master clock must be configured, by setting the SCLH and SCCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

A clock synchronization mechanism is implemented in order to support multi-master environment and slave clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a $t_{SYNC1}$ delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock.

I2C releases SCL to high level once the SCCLL counter reaches the value programmed in the SCCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a $t_{SYNC2}$ delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock.

I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the master clock period is:

$$t_{SCL} = t_{SYNC1} + t_{SYNC2} + [(SCLH + 1) + (SCCLL + 1)] \times (PRESC + 1) \times t_{I2CCLK}$$

The duration of $t_{SYNC1}$ depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): $DNF[3:0] \times t_{I2CCLK}$
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of $t_{SYNC2}$ depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): $DNF[3:0] \times t_{I2CCLK}$
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)
Figure 693. Master clock generation

SCL master clock generation

SCL master clock synchronization

SCL driven low by another device
Caution: For compliance with the I²C-bus or SMBus specification, the master clock must respect the timings in the following table.

### Table 527. I²C-bus and SMBus specification clock timings

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
<th>SMBus</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
<td>Max</td>
<td>Min</td>
</tr>
<tr>
<td>fSCL</td>
<td>SCL clock frequency</td>
<td>-</td>
<td>100</td>
<td>-</td>
<td>400</td>
<td>-</td>
</tr>
<tr>
<td>tHD:STA</td>
<td>Hold time (repeated) START condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tSU:STA</td>
<td>Set-up time for a repeated START condition</td>
<td>4.7</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tSU:STO</td>
<td>Set-up time for STOP condition</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tBUF</td>
<td>Bus free time between a STOP and START condition</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>tLOW</td>
<td>Low period of the SCL clock</td>
<td>4.7</td>
<td>-</td>
<td>1.3</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td>tHIGH</td>
<td>High period of the SCL clock</td>
<td>4.0</td>
<td>-</td>
<td>0.6</td>
<td>-</td>
<td>0.26</td>
</tr>
<tr>
<td>tR</td>
<td>Rise time of both SDA and SCL signals</td>
<td>-</td>
<td>1000</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
<tr>
<td>tF</td>
<td>Fall time of both SDA and SCL signals</td>
<td>-</td>
<td>300</td>
<td>-</td>
<td>300</td>
<td>-</td>
</tr>
</tbody>
</table>

Note: The SCL[7:0] bitfield also determines the tBUF and tSU:STA timings and SCLH[7:0] the tHD:STA and tSU:STO timings.

Refer to Section 51.4.10 for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

### Master communication initialization (address phase)

To initiate the communication with a slave to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: slave address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.
To launch the communication, set the START bit of the I2C_CR2 register. The master then automatically sends a START condition followed by the slave address as soon as it detects that the bus is free (BUSY = 0) and after the \( t_{\text{BUF}} \) delay from a previous STOP condition expires.

In case of an arbitration loss, the master automatically switches back to slave mode and can acknowledge its own address if it is addressed as a slave.

**Note:** The START bit is reset by hardware when the slave address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the master automatically keeps resending the slave address in a loop until the first address byte (first seven address bits) is acknowledged by the slave. Setting the ADDRCF bit makes I2C quit that loop.

If the I2C peripheral is addressed as a slave (ADDR = 1) while the START bit is set, the I2C peripheral switches to slave mode and the START bit is cleared.

**Note:** The same procedure is applied for a repeated start condition. In this case, BUSY = 1.

**Figure 694. Master initialization flow**

Initialization of a master receiver addressing a 10-bit address slave

If the slave address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the master automatically sends the following complete sequence after the START bit is set:

(RE)START + Slave address 10-bit header Write + Slave address second byte + (RE)START + Slave address 10-bit header Read.

**Figure 695. 10-bit address read access with HEAD10R = 0**
If the master addresses a 10-bit address slave, transmits data to this slave and then reads data from the same slave, a master transmission flow must be done first. Then a repeated START is set with the 10-bit slave address configured with HEAD10R = 1. In this case, the master sends this sequence:

RESTART + Slave address 10-bit header Read.

![Figure 696. 10-bit address read access with HEAD10R = 1](image)

**Master transmitter**

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper slave address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. the NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.
Figure 697. Transfer sequence flow for I2C master transmitter, N ≤ 255 bytes

Master transmission

Master initialization

NBYTES = N
AUTOEND = 0 for RESTART, 1 for STOP
Configure slave address
Set I2C_CR2.START

No

I2C_ISR.NACKF = 1?

Yes
End

No

I2C_ISR.TXIS = 1?

Yes
Write I2C_TXDR

No

NBYTES transmitted?

Yes

I2C_ISR.TC = 1?

Yes
Set I2C_CR2.START with slave addess NBYTES...

No

End

No

I2C_ISR.NACKF = 1?
Figure 698. Transfer sequence flow for I2C master transmitter, N > 255 bytes

Master transmission

Master initialization

NBYTES = 0xFF; N=N-255
RELOAD = 1
Configure slave address
Set I2C_CR2.START

I2C_ISR.NACKF = 1?
Yes
End

No

I2C_ISR.TXS = 1?
Yes
Write I2C_TXDR

No

NBYTES transmitted ?
Yes

I2C_ISR.TC = 1?
No

IF N= 256
NBYTES = N; N = 0; RELOAD = 0
AUTOEND = 0 for RESTART; 1 for STOP
ELSE
NBYTES = 0xFF; N = N-255
RELOAD = 1

Set I2C_CR2.START
with slave address
NBYTES _

Yes

No
Example I2C master transmitter 2 bytes, automatic end mode (STOP)

INIT: program Slave address, program NBYTES = 2, AUTOEND=1, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2

Example I2C master transmitter 2 bytes, software end mode (RESTART)

INIT: program Slave address, program NBYTES = 2, AUTOEND=0, set START
EV1: TXIS ISR: wr data1
EV2: TXIS ISR: wr data2
EV3: TC ISR: program Slave address, program NBYTES = N, set START
Master receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper slave address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and the START condition, followed by slave address, are sent on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.
Figure 700. Transfer sequence flow for I2C master receiver, N ≤ 255 bytes

Master initialization

NBYTES = N
AUTOEND = 0 for RESTART, 1 for STOP
Configure slave address
Set I2C_CR2.START

I2C_ISR.RXNE = 1?

No

Master reception

Yes

Read I2C_RXDR

NBYTES received?

No

Yes

I2C_ISR.TC = 1?

No

Set I2C_CR2.START with slave address NBYTES ...

Yes

End
Figure 701. Transfer sequence flow for I2C master receiver, N > 255 bytes

1. **Master initialization**

   - NBYTES = 0xFF; N=N-255
   - RELOAD = 1
   - Configure slave address
   - Set I2C_CR2.START

2. **I2C_ISR.RXNE = 1?**
   - No
   - Yes, Read I2C_RXDR

3. **NBYTES received?**
   - If N< 256
     - NBYTES = N; N=0; RELOAD=0
     - AUTOEND=0 for RESTART, 1 for STOP
   - ELSE
     - NBYTES = 0xFF; N=N-255
     - RELOAD=1

4. **I2C_ISR.TC = 1?**
   - No
   - Yes, Set I2C_CR2.START with slave address NBYTES...

5. **I2C_ISR.TCR = 1?**
   - No
   - Yes

6. **End**
51.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (I2C Configuration window).
### Table 528. Timing settings for fI2CCLK of 8 MHz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC[3:0]</td>
<td>0x1</td>
<td>0x1</td>
<td>0x0</td>
</tr>
<tr>
<td>SCLL[7:0]</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>tSCLL(1)</td>
<td>200 x 250 ns = 50 µs</td>
<td>20 x 250 ns = 5.0 µs</td>
<td>10 x 125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH[7:0]</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLH(1)</td>
<td>196 x 250 ns = 49 µs</td>
<td>16 x 250 ns = 4.0 µs</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
<tr>
<td>SDADEL[3:0]</td>
<td>0x2</td>
<td>0x2</td>
<td>0x1</td>
</tr>
<tr>
<td>tSDADEL</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 250 ns = 500 ns</td>
<td>1 x 125 ns = 125 ns</td>
</tr>
<tr>
<td>SCLDEL[3:0]</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLDEL</td>
<td>5 x 250 ns = 1250 ns</td>
<td>5 x 250 ns = 1250 ns</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. tSCL is greater than tSCLL + tSCLH due to SCL internal detection delay. Values provided for tSCL are examples only.
2. tSYNC1 + tSYNC2 minimum value is 4 x tI2CCLK = 500 ns. Example with tSYNC1 + tSYNC2 = 1000 ns.
3. tSYNC1 + tSYNC2 minimum value is 4 x tI2CCLK = 500 ns. Example with tSYNC1 + tSYNC2 = 750 ns.
4. tSYNC1 + tSYNC2 minimum value is 4 x tI2CCLK = 500 ns. Example with tSYNC1 + tSYNC2 = 655 ns.

### Table 529. Timing settings for fI2CCLK of 16 MHz

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Standard-mode (Sm)</th>
<th>Fast-mode (Fm)</th>
<th>Fast-mode Plus (Fm+)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10 kHz</td>
<td>100 kHz</td>
<td>400 kHz</td>
</tr>
<tr>
<td>PRESC[3:0]</td>
<td>0x3</td>
<td>0x3</td>
<td>0x1</td>
</tr>
<tr>
<td>SCLL[7:0]</td>
<td>0xC7</td>
<td>0x13</td>
<td>0x9</td>
</tr>
<tr>
<td>tSCLL(1)</td>
<td>200 x 250 ns = 50 µs</td>
<td>20 x 250 ns = 5.0 µs</td>
<td>10 x 125 ns = 1250 ns</td>
</tr>
<tr>
<td>SCLH[7:0]</td>
<td>0xC3</td>
<td>0xF</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLH(1)</td>
<td>196 x 250 ns = 49 µs</td>
<td>16 x 250 ns = 4.0 µs</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
<tr>
<td>tSCL(1)</td>
<td>~100 µs(2)</td>
<td>~10 µs(2)</td>
<td>~2.5 µs(3)</td>
</tr>
<tr>
<td>SDADEL[3:0]</td>
<td>0x2</td>
<td>0x2</td>
<td>0x2</td>
</tr>
<tr>
<td>tSDADEL</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 250 ns = 500 ns</td>
<td>2 x 125 ns = 250 ns</td>
</tr>
<tr>
<td>SCLDEL[3:0]</td>
<td>0x4</td>
<td>0x4</td>
<td>0x3</td>
</tr>
<tr>
<td>tSCLDEL</td>
<td>5 x 250 ns = 1250 ns</td>
<td>5 x 250 ns = 1250 ns</td>
<td>4 x 125 ns = 500 ns</td>
</tr>
</tbody>
</table>

1. tSCL is greater than tSCLL + tSCLH due to SCL internal detection delay. Values provided for tSCL are examples only.
2. tSYNC1 + tSYNC2 minimum value is 4 x tI2CCLK = 250 ns. Example with tSYNC1 + tSYNC2 = 1000 ns.
3. tSYNC1 + tSYNC2 minimum value is 4 x tI2CCLK = 250 ns. Example with tSYNC1 + tSYNC2 = 750 ns.
4. tSYNC1 + tSYNC2 minimum value is 4 x tI2CCLK = 250 ns. Example with tSYNC1 + tSYNC2 = 500 ns.
51.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Slave** is a device that receives or responds to a command.
- **Master** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized master that provides the main interface to the system CPU. A host must be a master-slave and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a master or a slave device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. A device can use any or all of them to communicate. The protocols are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in slave mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

A SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the slave byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Slave byte control mode* for more details.
Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a master and the host as a slave.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus slave device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus slave device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the C(x) = x^8 + x^2 + x + 1 CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

### Table 530. SMBus timeout specifications

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Limits</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>t TIMEOUT</em></td>
<td>Detect clock low timeout</td>
<td>25</td>
<td>35</td>
</tr>
<tr>
<td><em>t LOW:SEXT</em></td>
<td>Cumulative clock low extend time (slave device)</td>
<td>-</td>
<td>25</td>
</tr>
<tr>
<td><em>t LOW:MEXT</em></td>
<td>Cumulative clock low extend time (master device)</td>
<td>-</td>
<td>10</td>
</tr>
</tbody>
</table>

1. *t LOW:SEXT* is the cumulative time a given slave device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another slave device or the master also extends the clock causing the combined clock low extend time to be greater than *t LOW:SEXT*. Therefore, this parameter is measured with the slave device as the sole target of a full-speed master.
2. $t_{LOW:MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a slave device or another master also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. Therefore, this parameter is measured with a full speed slave device as the sole target of the master.

Figure 703. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$

Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH,MAX}$ (refer to I2C timings).

This timing parameter covers the condition where a master is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

51.4.12 SMBus initialization

In addition to the I2C initialization for the I2C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (slave mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in slave mode, the slave byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to Slave byte control mode for more details.

Specific addresses (slave mode)

The specific SMBus addresses must be enabled if required. Refer to Bus idle detection for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.
The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

**Packet error checking**

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in slave mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

**Caution:** Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

<table>
<thead>
<tr>
<th>Mode</th>
<th>SBC bit</th>
<th>RELOAD bit</th>
<th>AUTOEND bit</th>
<th>PECBYTE bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Tx/Rx NBYTES + PEC+ STOP</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Master Tx/Rx NBYTES + PEC + ReSTART</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Slave Tx/Rx with PEC</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

**Timeout detection**

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

**t_TIMEOUT check**

To check the t_TIMEOUT parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_I2CCLK period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to Table 532.

**Caution:** Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

**t_LOW:SEXT and t_LOW:MEXT check**

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking t_LOW:SEXT for the I2C peripheral operating as a slave, or t_LOW:MEXT when it operates as a master. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_I2CCLK period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.
Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the \( t_{\text{IDLE}} \) period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMEOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the \((\text{TIMEOUTA} + 1) \times 4 \times t_{\text{I2CCLK}}\) period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to Table 534.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

51.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach target \( t_{\text{TIMEOUT}}, t_{\text{LOW:SEXT}}, t_{\text{LOW:MEXT}}, \) and \( t_{\text{TIDLE}} \) timings at different \( f_{\text{I2CCLK}} \) frequencies.

Table 532. TIMEOUTA[11:0] for maximum \( t_{\text{TIMEOUT}} \) of 25 ms

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTA[11:0]</th>
<th>TIDLE</th>
<th>TIMEOUTEN</th>
<th>( t_{\text{TIMEOUT}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x61</td>
<td>0</td>
<td>1</td>
<td>( 98 \times 2048 \times 125 \text{ ns} = 25 \text{ ms} )</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC3</td>
<td>0</td>
<td>1</td>
<td>( 196 \times 2048 \times 62.5 \text{ ns} = 25 \text{ ms} )</td>
</tr>
</tbody>
</table>

Table 533. TIMEOUTB[11:0] for maximum \( t_{\text{LOW:SEXT}} \) and \( t_{\text{LOW:MEXT}} \) of 8 ms

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTB[11:0]</th>
<th>TEXTEN</th>
<th>( t_{\text{LOW:SEXT}} )</th>
<th>( t_{\text{LOW:MEXT}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x1F</td>
<td>1</td>
<td>( 32 \times 2048 \times 125 \text{ ns} = 8 \text{ ms} )</td>
<td></td>
</tr>
<tr>
<td>16 MHz</td>
<td>0x3F</td>
<td>1</td>
<td>( 64 \times 2048 \times 62.5 \text{ ns} = 8 \text{ ms} )</td>
<td></td>
</tr>
</tbody>
</table>

Table 534. TIMEOUTA[11:0] for maximum \( t_{\text{TIDLE}} \) of 50 \( \mu\text{s} \)

<table>
<thead>
<tr>
<th>( f_{\text{I2CCLK}} )</th>
<th>TIMEOUTA[11:0]</th>
<th>TIDLE</th>
<th>TIMEOUTEN</th>
<th>( t_{\text{TIDLE}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 MHz</td>
<td>0x63</td>
<td>1</td>
<td>1</td>
<td>( 100 \times 4 \times 125 \text{ ns} = 50 \mu\text{s} )</td>
</tr>
<tr>
<td>16 MHz</td>
<td>0xC7</td>
<td>1</td>
<td>1</td>
<td>( 200 \times 4 \times 62.5 \text{ ns} = 50 \mu\text{s} )</td>
</tr>
</tbody>
</table>

51.4.14 SMBus slave mode

In addition to I2C slave transfer management (refer to Section 51.4.8: I2C slave mode), this section provides extra software flowcharts to support SMBus.

SMBus slave transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit
is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the master requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 704. Transfer sequence flow for SMBus slave transmitter N bytes + PEC**

```plaintext
SMBus slave transmission

Slave initialization

No

I2C_ISR.ADDR = 1?

Yes

Read ADDCODE and DIR in I2C_ISR
I2C_CR2.NBYTES = N + 1
PECBYTE=1
Set I2C_ICR.ADDRCF

I2C_ISR.TXIS = 1?

No

SCL stretched

Yes

Write I2C_TXDR.TXDATA
```
**SMBus slave receiver**

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Slave byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

---

**Figure 705. Transfer bus diagram for SMBus slave transmitter (SBC = 1)**

Example SMBus slave transmitter 2 bytes + PEC,

![Transfer bus diagram](image)

<table>
<thead>
<tr>
<th>EV1</th>
<th>EV2</th>
<th>EV3</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDR</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>TXIS</td>
<td>data1</td>
<td>data2</td>
</tr>
<tr>
<td>TXIS</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NBYTES = 3

EV1: ADDR ISR: check ADDCODE, program NBYTES=3, set PECBYTE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2

**Legend:**
- transmission
- reception
- SCL stretch

**Caution:**
The PECBYTE bit has no effect when the RELOAD bit is set.
Figure 706. Transfer sequence flow for SMBus slave receiver N bytes + PEC

1. **SMBus slave reception**
2. **Slave initialization**
   - **I2C_ISR.ADDR = 1?**
     - No
     - Yes: Read ADDCODE and DIR in I2C_ISR
       - I2C_CR2.NBYTES = 1, RELOAD = 1
       - PECBYTE = 1
       - Set I2C_ICR.ADDRCF
3. **I2C_ISR.RXNE = 1? I2C_ISR.TCR = 1?**
   - Yes: Read I2C_RXDR.RXDATA
     - Program I2C_CR2.NACK = 0
     - I2C_CR2.NBYTES = 1
     - N = N - 1
   - No: Read I2C_RXDR.RXDATA
     - Program RELOAD = 0
     - NACK = 0 and NBYTES = 1
4. **N = 1?**
   - No: **I2C_ISR.RXNE = 1?**
     - No: End
     - Yes: Read I2C_RXDR.RXDATA
   - Yes: End

Note: SCL stretched
51.4.15 SMBus master mode

In addition to I2C master transfer management (refer to Section 51.4.9: I2C master mode), this section provides extra software flowcharts to support SMBus.

**SMBus master transmitter**

When the SMBus master wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus master wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.
When the SMBus master wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.

**Figure 708. Bus transfer diagrams for SMBus master transmitter**

**Example SMBus master transmitter 2 bytes + PEC, automatic end mode (STOP)**

INIT: program Slave address, program NBYTES = 3, AUTOEND=1, set PECBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

**Example SMBus master transmitter 2 bytes + PEC, software end mode (RESTART)**

INIT: program Slave address, program NBYTES = 3, AUTOEND=0, set PECBYTE, set START

EV1: TXIS ISR: wr data1

EV2: TXIS ISR: wr data2

EV3: TC ISR: program Slave address, program NBYTES = N, set START
**SMBus master receiver**

When the SMBus master wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the slave address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus master receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the slave address programmed before setting the START bit. In this case, after NBYTES - 1 data have been received, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

**Caution:** The PECBYTE bit has no effect when the RELOAD bit is set.
### 51.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI and CSI only oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI and CSI only oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI and CSI only oscillator and stretches SCL low until the oscillator wakes up.

HSI and CSI only is then used for the address reception.
If the received address matches the device own address, I2C stretches SCL low until the
device wakes up. The stretch is released when the ADDR flag is cleared by software. Then
the transfer goes on normally.

If the address does not match, the HSI and CSI only oscillator is stopped again and the
device does not wake up.

Note:  
When the system clock is used as I2C clock, or when WUPEN = 0, the HSI and CSI only
oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when
I2C is performing a transfer, either as a master or as an addressed slave after the ADDR
flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt
routine and setting it again only after the STOPF flag is set.

Caution:  The digital filter is not compatible with the wake-up from Stop mode feature. Before entering
Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the
DNF[3:0] bitfield.

Caution:  The feature is only available when the HSI and CSI only oscillator is selected as the I2C
clock.

Caution:  Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the
wake-up from Stop mode feature.

Caution:  If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be
disabled before entering Stop mode (PE = 0).

51.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

**Bus error (BERR)**

A bus error is detected when a START or a STOP condition is detected and is not located
after a multiple of nine SCL clock pulses. START or STOP condition is detected when an
SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as master or
addressed slave (that is, not during the address phase in slave mode).

In case of a misplaced START or RESTART detection in slave mode, the I2C peripheral
clears address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt
is generated if the ERRIE bit of the I2C_CR1 register is set.

**Arbitration loss (ARLO)**

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is
sampled on the SCL rising edge.

In master mode, arbitration loss is detected during the address phase, data phase and data
acknowledge phase. In this case, the SDA and SCL lines are released, the START control
bit is cleared by hardware and the master switches automatically to slave mode.

In slave mode, arbitration loss is detected during data phase and data acknowledge phase.
In this case, the transfer is stopped and the SCL and SDA lines are released.
When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Overrun/underrun error (OVR)**

An overrun or underrun error is detected in slave mode when NOSTRETCH = 1 and:
- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Packet error checking error (PECERR)**

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Timeout error (TIMEOUT)**

A timeout error occurs for any of these conditions:
- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect a bus idle condition.
- Master cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus tLOW:MEXT parameter).
- Slave cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus tLOW:SEXT parameter).

When a timeout violation is detected in master mode, a STOP condition is automatically sent.

When a timeout violation is detected in slave mode, SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

**Alert (ALERT)**

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.
51.5 I2C in low-power modes

Table 535. Effect of low-power modes to I2C

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. I2C interrupts cause the device to exit the Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The contents of I2C registers are kept.</td>
</tr>
<tr>
<td></td>
<td>– WUPEN = 1 and I2C is clocked by an internal oscillator (HSI and CSI only). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode.</td>
</tr>
<tr>
<td></td>
<td>– WUPEN = 0: the I2C must be disabled before entering Stop mode.</td>
</tr>
<tr>
<td>Standby</td>
<td>The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 51.3: I2C implementation for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

51.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 536. I2C interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop modes</th>
<th>Exit Standby modes</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_EV</td>
<td>Receive buffer not empty</td>
<td>RXNE</td>
<td>RXIE</td>
<td>Read I2C_RXDR register</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit buffer interrupt status</td>
<td>TXIS</td>
<td>TXIE</td>
<td>Write I2C_TXDR register</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>STOP detection interrupt flag</td>
<td>STOPF</td>
<td>STOPIE</td>
<td>Write STOPCF = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transfer complete reload</td>
<td>TCR</td>
<td>TCIE</td>
<td>Write I2C_CR2 with NBYTES[7:0] ≠ 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transfer complete</td>
<td>TC</td>
<td></td>
<td>Write START = 1 or STOP = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Address matched</td>
<td>ADDR</td>
<td>ADDRIE</td>
<td>Write ADDRCF = 1</td>
<td>Yes(1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NACK reception</td>
<td>NACKF</td>
<td>NACKIE</td>
<td>Write NACKCF = 1</td>
<td></td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>I2C_ER</td>
<td>Bus error</td>
<td>BERR</td>
<td>ERRIE</td>
<td>Write BERRCF = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Arbitration loss</td>
<td>ARLO</td>
<td></td>
<td>Write ARLOCF = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun/underrun</td>
<td>OVR</td>
<td></td>
<td>Write OVRCF = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I2C_ER</td>
<td>PEC error</td>
<td>PECERR</td>
<td>ERRIE</td>
<td>Write PECERRCF = 1</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Timeout/ t_LOW error</td>
<td>TIMEOUT</td>
<td></td>
<td>Write TIMEOUTCF = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SMBus alert</td>
<td>ALERT</td>
<td></td>
<td>Write ALERTCF = 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The ADDR match event can wake up the device from Stop mode only if the I2C I2C instance supports the wake-up from Stop mode feature. Refer to Section 51.3: I2C implementation.
51.7 I2C DMA requests

Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see Section 12: General purpose direct memory access controller (GPDMA)) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software (the transmitted slave address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to Master transmitter.

In slave mode:

• With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
• With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to SMBus slave transmitter and SMBus master mode.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to Section 12: General purpose direct memory access controller (GPDMA)) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In master mode, the initialization, the slave address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In slave mode with NOSTRETCH = 0, when all data are transferred using DMA, the DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to SMBus slave receiver and SMBus master receiver.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

51.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the I2Cx bits in the Debug infrastructure block.
51.9  I2C registers

Refer to Section 1.2 for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

51.9.1  I2C control register 1 (I2C_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

|-----------|----------|------|------|------|------|-----|-------|-------|------|------|----|-----|-----------|-----|

Bit 31  STOPFACLR: STOP detection flag (STOPF) automatic clear
0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.
1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH slave mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bit 30  ADDRACLR: Address match flag (ADDR) automatic clear
0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.
1: ADDR flag remains cleared by hardware. This mode can be used in slave mode, to avoid the ADDR clock stretching if the I2C enables only one slave address. This allows a slave data management by DMA only, without any interrupt from peripheral.

Bits 29:25  Reserved, must be kept at reset value.

Bit 24  FMP: Fast-mode Plus 20 mA drive enable
0: 20 mA I/O drive disabled
1: 20 mA I/O drive enabled

Bit 23  PECEN: PEC enable
0: PEC calculation disabled
1: PEC calculation enabled
Bit 22  **ALERTEN**: SMBus alert enable
0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).
*Note*: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21  **SMBDEN**: SMBus device default address enable
0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20  **SMBHEN**: SMBus host address enable
0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19  **GCEN**: General call enable
0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18  **WUPEN**: Wake-up from Stop mode enable
0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.
*Note*: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17  **NOSTRETCH**: Clock stretching disable
This bit is used to disable clock stretching in slave mode. It must be kept cleared in master mode.
0: Clock stretching enabled
1: Clock stretching disabled
*Note*: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16  **SBC**: Slave byte control
This bit is used to enable hardware byte control in slave mode.
0: Slave byte control disabled
1: Slave byte control enabled

Bit 15  **RXDMAEN**: DMA reception requests enable
0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14  **TXDMAEN**: DMA transmission requests enable
0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13  Reserved, must be kept at reset value.

Bit 12  **ANOFF**: Analog noise filter OFF
0: Analog noise filter enabled
1: Analog noise filter disabled
*Note*: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).
**Bits 11:8**  
**DNF[3:0]: Digital noise filter**  
These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter filters spikes with a length of up to $DNF[3:0] \times t_{I2CCLK}$  
0000: Digital filter disabled  
0001: Digital filter enabled and filtering capability up to one $t_{I2CCLK}$  
...  
1111: digital filter enabled and filtering capability up to fifteen $t_{I2CCLK}$  
*Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).*

**Bit 7**  
**ERRIE: Error interrupts enable**  
0: Error detection interrupts disabled  
1: Error detection interrupts enabled  
*Note: Any of these errors generates an interrupt:*  
- arbitration loss (ARLO)  
- bus error detection (BERR)  
- overrun/underrun (OVR)  
- timeout detection (TIMEOUT)  
- PEC error detection (PECERR)  
- alert pin event detection (ALERT)

**Bit 6**  
**TCIE: Transfer complete interrupt enable**  
0: Transfer complete interrupt disabled  
1: Transfer complete interrupt enabled  
*Note: Any of these events generates an interrupt:*  
Transfer complete (TC)  
Transfer complete reload (TCR)

**Bit 5**  
**STOPIE: STOP detection interrupt enable**  
0: STOP detection (STOPF) interrupt disabled  
1: STOP detection (STOPF) interrupt enabled

**Bit 4**  
**NACKIE: Not acknowledge received interrupt enable**  
0: Not acknowledge (NACKF) received interrupts disabled  
1: Not acknowledge (NACKF) received interrupts enabled

**Bit 3**  
**ADDRIE: Address match interrupt enable (slave only)**  
0: Address match (ADDR) interrupts disabled  
1: Address match (ADDR) interrupts enabled

**Bit 2**  
**RXIE: RX interrupt enable**  
0: Receive (RXNE) interrupt disabled  
1: Receive (RXNE) interrupt enabled

**Bit 1**  
**TXIE: TX interrupt enable**  
0: Transmit (TXIS) interrupt disabled  
1: Transmit (TXIS) interrupt enabled

**Bit 0**  
**PE: Peripheral enable**  
0: Peripheral disabled  
1: Peripheral enabled  
*Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.*
51.9.2  I2C control register 2 (I2C_CR2)

Address offset: 0x04
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td><strong>PECBYTE</strong>: Packet error checking byte</td>
</tr>
<tr>
<td></td>
<td>This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.</td>
</tr>
<tr>
<td></td>
<td>0: No PEC transfer</td>
</tr>
<tr>
<td></td>
<td>1: PEC transmission/reception is requested</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Writing 0 to this bit has no effect.</td>
</tr>
<tr>
<td></td>
<td>This bit has no effect when RELOAD is set, and in slave mode when SBC = 0.</td>
</tr>
<tr>
<td>25</td>
<td><strong>AUTOEND</strong>: Automatic end mode (master mode)</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.</td>
</tr>
<tr>
<td></td>
<td>1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This bit has no effect in slave mode or when the RELOAD bit is set.</td>
</tr>
<tr>
<td>24</td>
<td><strong>RELOAD</strong>: NBYTES reload mode</td>
</tr>
<tr>
<td></td>
<td>This bit is set and cleared by software.</td>
</tr>
<tr>
<td></td>
<td>0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).</td>
</tr>
<tr>
<td></td>
<td>1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.</td>
</tr>
<tr>
<td>23:16</td>
<td><strong>NBYTES[7:0]</strong>: Number of bytes</td>
</tr>
<tr>
<td></td>
<td>The number of bytes to be transmitted/received is programmed there. This field is don’t care in slave mode with SBC = 0.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Changing these bits when the START bit is set is not allowed.</td>
</tr>
</tbody>
</table>
Bit 15 **NACK**: NACK generation (slave mode)
The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.
0: an ACK is sent after current received byte.
1: a NACK is sent after current received byte.
*Note:* Writing 0 to this bit has no effect.

This bit is used only in slave mode: in master receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in slave receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation
This bit only pertains to master mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.
0: No STOP generation
1: STOP generation after current byte transfer
*Note:* Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation
This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in slave mode, by a timeout error detection, or when PE = 0.
0: No START generation
1: RESTART/START generation:
If the I2C is already in master mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer. Otherwise, setting this bit generates a START condition once the bus is free.
*Note:* Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in slave mode.
This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (master receiver mode)
0: The master sends the complete 10-bit slave address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.
1: The master sends only the first seven bits of the 10-bit address, followed by read direction.
*Note:* Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (master mode)
0: The master operates in 7-bit addressing mode
1: The master operates in 10-bit addressing mode
*Note:* Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (master mode)
0: Master requests a write transfer
1: Master requests a read transfer
*Note:* Changing this bit when the START bit is set is not allowed.
51.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, write states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 15</td>
<td><strong>OA1EN</strong>: Own address 1 enable</td>
</tr>
<tr>
<td></td>
<td>0: Own address 1 disabled. The received slave address OA1 is NACKed.</td>
</tr>
<tr>
<td></td>
<td>1: Own address 1 enabled. The received slave address OA1 is ACKed.</td>
</tr>
<tr>
<td>Bit 14:11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 10</td>
<td><strong>OA1MODE</strong>: Own address 1 10-bit mode</td>
</tr>
<tr>
<td></td>
<td>0: Own address 1 is a 7-bit address.</td>
</tr>
<tr>
<td></td>
<td>1: Own address 1 is a 10-bit address.</td>
</tr>
<tr>
<td>Note:</td>
<td><em>This bit can be written only when OA1EN = 0.</em></td>
</tr>
</tbody>
</table>

| Bits 9:0  | Interface own slave address              |
| 7-bit addressing mode: OA1[7:1] contains the 7-bit own slave address. Bits OA1[9], OA1[8] and OA1[0] are don’t care. |
| 10-bit addressing mode: OA1[9:0] contains the 10-bit own slave address. |
| Note:     | *These bits can be written only when OA1EN = 0.* |

51.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed.
completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable

0: Own address 2 disabled. The received slave address OA2 is NACKed.
1: Own address 2 enabled. The received slave address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks

000: No mask
001: OA2[1] is masked and don’t care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don’t care. Only OA2[7:3] are compared.
100: OA2[4:1] are masked and don’t care. Only OA2[7:5] are compared.
111: OA2[7:1] are masked and don’t care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.

**Note**: These bits can be written only when OA2EN = 0.

As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address

7-bit addressing mode: 7-bit address

**Note**: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

### 51.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCLH[7:0]</td>
<td>SCLI[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

RM0477 Rev 6 2289/3791
51.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14
Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>PRESC[3:0]: Timing prescaler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 27:24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bits 23:20</td>
<td>SCLDEL[3:0]: Data hold time</td>
</tr>
<tr>
<td>Bits 19:16</td>
<td>SDADEL[3:0]: Data setup time</td>
</tr>
<tr>
<td>Bits 15:8</td>
<td>SCLH[7:0]: SCL high period (master mode)</td>
</tr>
<tr>
<td>Bits 7:0</td>
<td>SCLL[7:0]: SCL low period (master mode)</td>
</tr>
</tbody>
</table>

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.
Bit 31 TEXTEN: Extended clock timeout enable
   0: Extended clock timeout detection is disabled
   1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more
   than t_{LOW:EXT} is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 TIMEOUTB[11:0]: Bus timeout B
   This field is used to configure the cumulative clock extension timeout:
   – Master mode: the master cumulative clock low extend time (t_{LOW:MEXT}) is detected
   – Slave mode: the slave cumulative clock low extend time (t_{LOW:SEXT}) is detected

\[ t_{LOW:EXT} = (\text{TIMEOUTB} + \text{TIDLE} = 01) \times 2048 \times \text{f}_{\text{I2CCLK}} \]

Note: These bits can be written only when TEXTEN = 0.

Bit 15 TIMOUTEN: Clock timeout enable
   0: SCL timeout detection is disabled
   1: SCL timeout detection is enabled. When SCL is low for more than t_{TIMEOUT} (TIDLE = 0) or
   high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 TIDLE: Idle clock timeout detection
   0: TIMEOUTA is used to detect SCL low timeout
   1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 TIMEOUTA[11:0]: Bus timeout A
   This field is used to configure:
   The SCL low timeout condition t_{TIMEOUT} when TIDLE = 0
   \[ t_{\text{TIMEOUT}} = (\text{TIMEOUTA} + 1) \times 2048 \times \text{f}_{\text{I2CCLK}} \]
   The bus idle condition (both SCL and SDA high) when TIDLE = 1
   \[ t_{\text{IDLE}} = (\text{TIMEOUTA} + 1) \times 4 \times \text{f}_{\text{I2CCLK}} \]

Note: These bits can be written only when TIMOUTEN = 0.

51.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18
Reset value: 0x0000 0001
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADDCODE[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>BUSY</td>
<td>Ready</td>
<td>ALERT</td>
<td>TIME OUT</td>
<td>PEC</td>
<td>ERR</td>
<td>OVR</td>
<td>ARLO</td>
<td>BERR</td>
<td>TCR</td>
<td>TC</td>
<td>STOPF</td>
<td>NACKF</td>
<td>ADDR</td>
<td>RXNE</td>
<td>TXIS</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rs</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.
Bits 23:17  **ADDCODE[6:0]**: Address match code (slave mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

**Bit 16**  **DIR**: Transfer direction (slave mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, slave enters receiver mode.

1: Read transfer, slave enters transmitter mode.

**Bit 15**  **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

**Bit 14**  Reserved, must be kept at reset value.

**Bit 13**  **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

*Note:* This bit is cleared by hardware when PE = 0.

**Bit 12**  **TIMEOUT**: Timeout or tLOW detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

*Note:* This bit is cleared by hardware when PE = 0.

**Bit 11**  **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

*Note:* This bit is cleared by hardware when PE = 0.

**Bit 10**  **OVR**: Overrun/underrun (slave mode)

This flag is set by hardware in slave mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

*Note:* This bit is cleared by hardware when PE = 0.

**Bit 9**  **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

*Note:* This bit is cleared by hardware when PE = 0.

**Bit 8**  **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in slave mode. It is cleared by software by setting the BERRCF bit.

*Note:* This bit is cleared by hardware when PE = 0.

**Bit 7**  **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

*Note:* This bit is cleared by hardware when PE = 0.

This flag is only for master mode, or for slave mode when the SBC bit is set.
Bit 6  **TC**: Transfer complete (master mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 5  **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a master, provided that the STOP condition is generated by the peripheral.
- as a slave, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 4  **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 3  **ADDR**: Address matched (slave mode)

This bit is set by hardware as soon as the received slave address matched with one of the enabled slave addresses. It is cleared by software by setting ADDRCF bit.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 2  **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

*Note: This bit is cleared by hardware when PE = 0.*

Bit 1  **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

*Note: This bit is cleared by hardware when PE = 0.*

Bit 0  **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

*Note: This bit is set by hardware when PE = 0.*
51.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>Bit 31:14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 13</td>
<td>ALERTCF: Alert flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 12</td>
<td>TIMOUTCF: Timeout detection flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 11</td>
<td>PECCF: PEC error flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 10</td>
<td>OVRCF: Overrun/underrun flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the OVR flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 9</td>
<td>ARLOCF: Arbitration lost flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 8</td>
<td>BERRCF: Bus error flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bits 7:6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>STOPCF: STOP detection flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>NACKCF: Not acknowledge flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the NACKF flag in I2C_ISR register.</td>
</tr>
<tr>
<td>Bit 3</td>
<td>ADDRCF: Address matched flag clear</td>
</tr>
<tr>
<td></td>
<td>Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.</td>
</tr>
<tr>
<td>Bits 2:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
51.9.9  I2C PEC register (I2C_PECR)

Address offset: 0x20
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PEC[7:0]**: Packet error checking register

This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

51.9.10  I2C receive data register (I2C_RXDR)

Address offset: 0x24
Reset value: 0x0000 0000
Access: no wait states

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **RXDATA[7:0]**: 8-bit receive data

Data byte received from the I²C-bus.
51.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28
Reset value: 0x0000 0000
Access: no wait states

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

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data
Data byte to be transmitted to the I²C-bus

*Note: These bits can be written only when TXE = 1.*
### 51.9.12 I2C register map

The table below provides the I2C register map and the reset values.

**Table 537. I2C register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>I2C_CR1</td>
<td>31</td>
<td>STOPF</td>
<td>30</td>
<td>ADDRCLR</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>FMP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>NBYTES[7:0]</td>
<td></td>
<td>PECEN</td>
<td></td>
<td>ALERTEN</td>
<td></td>
<td>SMBIDEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SMBHEN</td>
<td></td>
<td>GZHEN</td>
<td></td>
<td>WUPEN</td>
<td></td>
<td>NOSTRETCH</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RXMAEN</td>
<td></td>
<td>ANF[7:0]</td>
<td></td>
<td>DNF[3:0]</td>
<td></td>
<td>STOPF</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x04</td>
<td>I2C_CR2</td>
<td>31</td>
<td>RXDMAEN</td>
<td>30</td>
<td>TXDMAEN</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x08</td>
<td>I2C_OAR1</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x0C</td>
<td>I2C_OAR2</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x10</td>
<td>I2C_TIMINGR</td>
<td>31</td>
<td>DAR[7:0]</td>
<td>30</td>
<td>SAR[7:0]</td>
<td>29</td>
<td>SAR[7:0]</td>
<td>28</td>
<td>SAR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SAR[7:0]</td>
<td></td>
<td>SAR[7:0]</td>
<td></td>
<td>SAR[7:0]</td>
<td></td>
<td>SAR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x14</td>
<td>I2C_TIMEOUTR</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x18</td>
<td>I2C_ISR</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x1C</td>
<td>I2C_ICR</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x20</td>
<td>I2C_PECR</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x24</td>
<td>I2C_RXDR</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x28</td>
<td>I2C_TXDR</td>
<td>31</td>
<td>ADDR[7:0]</td>
<td>30</td>
<td>ADDR[7:0]</td>
<td>29</td>
<td>ADDR[7:0]</td>
<td>28</td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
<td></td>
<td>ADDR[7:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
52 Improved inter-integrated circuit (I3C)

52.1 Introduction

The I3C interface handles communication between this device and others, such as sensors and host processor, connected on an I3C bus.

An I3C bus is a two-wire, serial single-ended, multidrop bus, intended to improve a legacy I²C bus.

The I3C SDR-only peripheral implements all the features required by the MIPI® I3C specification v1.1. It can control all I3C bus-specific sequencing, protocol, arbitration, and timing, and can act as controller (formerly known as master), or as target (formerly known as slave).

When acting as controller, the I3C peripheral improves the features of the I²C interface preserving some backward compatibility: it allows an I²C target to operate on an I3C bus in legacy I²C fast-mode (Fm) or legacy I²C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C peripheral can be used with DMA, to off-load the CPU.

52.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in Table 541), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I²C read/write messages to legacy I²C targets in Fm/Fm+
  - I3C SDR read/write private messages
  - I3C SDR broadcast CCC messages (see details in Table 547)
  - I3C SDR read/write direct CCC messages (see details in Table 547)
- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with $x = 0, 1, 2, 3$ when controller
  - TEx with $x = 0, 1, \ldots, 6$ when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see Section 52.3.2):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see Section 52.3.2):
  - On a reset pattern
  - On a missed start
- Multiclock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in Section 52.6.2)
52.3  I3C implementation

52.3.1  I3C instantiation

There is a single I3C instance in the device.

52.3.2  I3C wake-up from low-power mode(s)

The peripheral can wake up the device from a low-power mode, as detailed in Table 538. For more details about the wake-up capabilities, refer to Section 52.13.

<table>
<thead>
<tr>
<th>Wake-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Stop mode with SVOS high</td>
</tr>
</tbody>
</table>

52.3.3  I3C FIFOs

The FIFOs are implemented as defined in Table 539.

<table>
<thead>
<tr>
<th>FIFO</th>
<th>Content</th>
<th>Unit</th>
<th>Size (in unit)</th>
<th>Used as controller/target (rationale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C-FIFO</td>
<td>(32-bit) Control words</td>
<td>Word</td>
<td>2</td>
<td>Controller (a frame can be based on multiple control words; this is not the case as target)</td>
</tr>
<tr>
<td>S-FIFO</td>
<td>(32-bit) Status words</td>
<td></td>
<td></td>
<td>Controller (target: status only in register mode)</td>
</tr>
<tr>
<td>TX-FIFO</td>
<td>Transmitted data</td>
<td>Byte</td>
<td>8</td>
<td>Controller and target</td>
</tr>
<tr>
<td>RX-FIFO</td>
<td>Received data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

52.3.4  I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

52.3.5  I3C interrupt(s)

The interrupt mapping is implemented as detailed in Table 540.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3c_err_it</td>
<td>O</td>
<td>Error interrupt line</td>
</tr>
<tr>
<td>i3c_evt_it</td>
<td>O</td>
<td>Event interrupt line</td>
</tr>
</tbody>
</table>
52.3.6 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in Table 541.

Table 541. I3C peripheral controller/target features versus MIPI v1.1

<table>
<thead>
<tr>
<th>Feature</th>
<th>MIPI I3C v1.1</th>
<th>I3C peripheral</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C SDR message</td>
<td>X</td>
<td>X X X</td>
<td>Mandatory when controller, and the I3C bus is mixed with (external) legacy (i^2C) target(s). Optional in MIPI v1.1 when target.</td>
</tr>
<tr>
<td>Legacy (i^2C) message (Fm/Fm+)</td>
<td>X</td>
<td>X -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>HDR DDR message</td>
<td>X</td>
<td>- -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>HDR-TSL/TSP, HDR-BT</td>
<td>X</td>
<td>- -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>Dynamic address assignment</td>
<td>X</td>
<td>X X X</td>
<td>-</td>
</tr>
<tr>
<td>Static address</td>
<td>X</td>
<td>X -</td>
<td>No (intended) support of the peripheral as a target on an (i^2C) bus.</td>
</tr>
<tr>
<td>Grouped addressing</td>
<td>X</td>
<td>X -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>CCCs</td>
<td>X</td>
<td>X X X</td>
<td>Mandatory and some optional CCCs supported (refer to Table 547 when controller/target).</td>
</tr>
<tr>
<td>Error detection and recovery</td>
<td>X</td>
<td>X X X</td>
<td>-</td>
</tr>
<tr>
<td>In-band interrupt (with MDB)</td>
<td>X</td>
<td>X X X</td>
<td>-</td>
</tr>
<tr>
<td>Secondary controller</td>
<td>X</td>
<td>X X X</td>
<td>-</td>
</tr>
<tr>
<td>Hot-join mechanism</td>
<td>X</td>
<td>X X X</td>
<td>-</td>
</tr>
<tr>
<td>Target reset</td>
<td>X</td>
<td>X X X</td>
<td>-</td>
</tr>
<tr>
<td>Synchronous timing control</td>
<td>X</td>
<td>X -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>Asynchronous timing control 0</td>
<td>X</td>
<td>X -</td>
<td>Mandatory in MIPI v1.1 when controller. Optional in MIPI v1.1 when target.</td>
</tr>
<tr>
<td>Asynchronous timing control 1, 2, 3</td>
<td>X</td>
<td>- -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>Device to device tunneling</td>
<td>X</td>
<td>X -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>Multi-lane data transfer</td>
<td>X</td>
<td>- -</td>
<td>Optional in MIPI v1.1</td>
</tr>
<tr>
<td>Monitoring device early termination</td>
<td>X</td>
<td>- -</td>
<td>Optional in MIPI v1.1</td>
</tr>
</tbody>
</table>
52.4 I3C block diagram

The I3C block diagram is illustrated in Figure 710.

Figure 710. I3C block diagram

1. This feature is implementation-dependent, and can be unavailable. Refer to Section 52.3.4: I3C triggers.

52.5 I3C pins and internal signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_SDA</td>
<td>Bidirectional</td>
<td>I3C bus serial data line</td>
</tr>
<tr>
<td>I3C_SCL</td>
<td>Bidirectional</td>
<td>I3C bus serial clock line</td>
</tr>
</tbody>
</table>

Table 542. I3C input/output pins

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3c_pclk</td>
<td>I</td>
<td>APB clock</td>
</tr>
<tr>
<td>i3c_ker_clk</td>
<td>I</td>
<td>Kernel clock (also named as I3CCLK)</td>
</tr>
<tr>
<td>i3c_pclk_req</td>
<td>O</td>
<td>APB clock request</td>
</tr>
<tr>
<td>i3c_ker_clk_req</td>
<td>O</td>
<td>Kernel clock request</td>
</tr>
<tr>
<td>i3c_it(1)</td>
<td>O</td>
<td>Global interrupt line</td>
</tr>
</tbody>
</table>

Table 543. I3C internal input/output signals
52.6 **I3C reset and clocks**

52.6.1 **I3C reset**

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to Table 547 and Section 52.16.16 for more details.

This reset interrupt notification can be used to wake up from a low power mode, where the I3C peripheral (typically in the VCore domain) is active.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

52.6.2 **I3C clocks and requirements**

As indicated in the **Figure 710**, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming **I3C timing register 0** (I3C_TIMINGR0), **I3C timing register 1** (I3C_TIMINGR1) and **I3C timing register 2** (I3C_TIMINGR2), as summarized in **Controller initialization** and **Updating the configuration for a transfer, as controller**.
  - When target: the user must set and comply with the bus available condition ($t_{AVAL}$ for an in-band interrupt or controller-role request), and the bus idle condition ($t_{IDLE}$ for a hot-join request), by programming **I3C timing register 1** (I3C_TIMINGR1), as summarized in **Target initialization**.

---

**Table 543. I3C internal input/output signals (continued)**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>i3c_err_it(1)</td>
<td>O</td>
<td>Error interrupt line</td>
</tr>
<tr>
<td>i3c_evt_it(1)</td>
<td>O</td>
<td>Event interrupt line</td>
</tr>
<tr>
<td>i3c_rx_dma</td>
<td>O</td>
<td>DMA request for reading received bytes/words from RX-FIFO</td>
</tr>
<tr>
<td>i3c_tx_dma</td>
<td>O</td>
<td>DMA request for writing to be transmitted bytes/words to TX-FIFO</td>
</tr>
<tr>
<td>i3c_tc_dma</td>
<td>O</td>
<td>DMA request for writing to be transmitted control words to C-FIFO, when peripheral acts as controller</td>
</tr>
<tr>
<td>i3c_rs_dma</td>
<td>O</td>
<td>DMA request for reading status words from S-FIFO, when peripheral acts as controller</td>
</tr>
</tbody>
</table>

1. This signal is implementation-dependent. Refer to Section 52.3.5.
- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to Section 7: Reset and clock control (RCC)).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock.

Note: Sustaining \( F_{SCL,max} = 12.9 \text{ MHz} \) means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < \( t_{HIGH} \) (SCL clock high period)
   - \( t_{HIGH,min} = 24 \text{ ns} \). A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller

2. Period of the I3CCLK kernel clock < \( t_{CASr} \) (clock after repeated start condition)
   - \( t_{CASr,min} = t_{CAS \, min} / 2 = 19.2 \text{ ns} \). A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller

3. Two periods of the I3CCLK kernel clock < \( t_{LOW \_OD} \) (SCL clock low period in open drain)
   - \( t_{LOW \_OD,min} = 200 \text{ ns} \). A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller

4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - \( F_{SCL,max} = 12.9 \text{ MHz} \). A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

- APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that \( F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3 \times F_{I3CCLK} - F_{SCL})] \)

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.
52.7 I3C peripheral state and programming

52.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see Figure 711 and Figure 712), the peripheral is in one of the following states:

- **Disabled state:**
  - After an I3C reset (system reset or I3C reset from RCC), the peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the peripheral takes into account the value of the different configuration registers, and switches to the (enabled and) idle state.

- **Idle state:**
  - After being enabled (EN = 1), the peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see Updating the configuration for a transfer, as controller and Updating the configuration of the I3C peripheral, as target for more details.
  - The peripheral switches to the (enabled and) active state, either:
    a) Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request)
    b) Or once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).

- **Active state:**
  - The peripheral executes the transfer(s) on the bus
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the I3C event register (I3C_EVR), and the corresponding interrupt is enabled by I3C interrupt enable register (I3C_IER). The peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured.
    a) As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF)
    b) As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF)

**Note:** The software can disable the peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

52.7.2 I3C controller state and programming sequence

*Figure 711* illustrates the overall programming sequence of the peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.
1. This feature is implementation-dependent and can be unavailable. Refer to Section 52.3.4: I3C triggers.
Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable

- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
     - SCL clock high time period \( t_{DIG_H}, t_{DIG_H MIXED} \) in legacy \( i^2C \) and I3C open-drain/push-pull
     - SCL clock low time period \( t_{DIG_L}, t_{DIG_OD_L} \) in legacy \( i^2C \) and I3C open-drain/push-pull
  b) *I3C timing register 1 (I3C_TIMINGR1)*:
     - SDA hold time in push-pull \( t_{HD_PP} \)
     - bus free condition time \( (I3C t_{CAS}, legacy i^2C t_{BUF}) \)
     - I3C repeated start timing \( t_{CAS}, t_{CBS} \)
     - I3C stop timing \( t_{CBP} \)
     - SCL clock low maximum stalling on the ENTDAA CCC \( (t_{STALLDAA}) \), or on the ACK/NACK of a legacy \( i^2C \) transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy \( i^2C \) write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy \( i^2C \) write \( t_{STALL} \), to adjust SCL clock low stalling, if needed by the peripheral itself, when used as controller
     - \( t_{NEWCRLOCK} \) for controller-role hand-off procedure (after GETACCCR CCC)
  c) *I3C timing register 2 (I3C_TIMINGR2)*:
     - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy \( i^2C \) target

- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*

- Configure the management of any device target \( x \): *I3C device x characteristics register (I3C_DEVRx)*, for \( x = 1 \) to \( x \leq 4 \)

- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable

- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*
Then, the software can enable the I3C peripheral (set EN = 1)

**Note:** The software can write once all the fields of the I3C_CFGR while enabling it.

### Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

1. **Software-triggering:** on a write and set TSFSET = 1 in the I3C_CFGR register
   - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
2. **No triggering:** on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

   a) If the C-FIFO is not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)

   b) If the C-FIFO is configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

### Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum \(t_{CAS}\) time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in Section 52.9:

- **Figure 723:** IBI transfer, as controller/target
- **Figure 724:** Hot-join request transfer, as controller/target
- **Figure 725:** Controller-role request transfer, as controller/target

### Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.
For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in Section 52.9:

- **Figure 713**: I3C CCC messages, as controller
- **Figure 714**: I3C broadcast ENTDAA CCC, as controller
- **Figure 715**: I3C broadcast, direct read and direct write RSTACT CCC, as controller
- **Figure 720**: I3C private read/write messages, as controller
- **Figure 722**: Legacy I2C read/write messages, as controller

*Figure 711* does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in Section 52.10.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating **I3C status register (I3C_SR)**, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to Section 52.10.4.

### Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via **I3C timing register 2 (I3C_TIMINGR2)**
- Modify the interrupt/polling mode policy via **I3C interrupt enable register (I3C_IER)**
- Modify the following fields of the **I3C configuration register (I3C_CFGR)**:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH
• Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  – \textit{I3C message control register (I3C\_CR), I3C message control register [alternate] (I3C\_CR)}
  – \textit{I3C status register (I3C\_SR)}
  – \textit{I3C transmit data byte register (I3C\_TDR), I3C transmit data word register (I3C\_TDWR)}
  – \textit{I3C receive data byte register (I3C\_RDR), I3C receive data word register (I3C\_RDWR)}

• Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  – Modify the hot-join acknowledge policy via bit HJACK in the I3C\_CFGR register
  – Modify IBI/CR acknowledge policy to any target x, via \textit{I3C device x characteristics register (I3C\_DEVRx)}

The registers usage vs. the I3C peripheral role as controller is summarized in Section 52.8.1.

The static/dynamic registers fields usage when acting as controller is summarized in Table 545.

52.7.3 I3C target state and programming sequence

\textit{Figure 712} illustrates the overall programming sequence of the peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.
Figure 712. I3C target state and programming sequence diagram

I3C state = IDLE

Prepare IBI payload if any: write I3C_IBIDR if I3C_BCR.BCR2=1

Receiving a (write) broadcast CCC on the I3C bus

Initiate the IBI request: write I3C_CR (with MTYPE[3:0]=1010)

Transmitting IBI payload data if any (if I3C_BCR.BCR2=1)

Setting IBIENDF = 1

IBI completed

No transfer error?

Setting ERRF = 1

Transfer error

Executing the IBI request on the I3C bus

Initiate the controller-role request: write I3C_CR (with MTYPE[3:0]=1001)

Setting CRUPDF = 1

Control gaining completed

GETACCCR received and completed?

Setting ERRF = 1

Transfer error

Executing the CR request on the I3C bus

Initiate the hot-join request: write I3C_CR (with MTYPE[3:0]=1000)

Setting DAUPDF = 1

Dynamic addressing assignment completed

ENTDAA received and completed?

Setting ERRF = 1

Transfer error

Executing the HJ request on the I3C bus

Acknowledge? (by controller)

Executing CCC specific action

Updating related registers including I3C_SR

No transfer error?

Setting ERRF = 1

Transfer error

Setting xxUPDF=1 or RSTF=1

GRPF/DEFF=1

CCC completed

Receiving and acknowledging a direct read/write CCC on the I3C bus

Executing CCC specific action

Updating/using related registers including I3C_SR

No transfer error?

Setting ERRF = 1

Transfer error

Setting I3C_EVR.xxUPDF=1 or I3C_EVR.GETF/STAF=1

CCC completed

If CCC= (DEFGRPA or DEFTGTS):

Receiving write data byte(s) into I3C_RD(W)R

Configure/prepare I3C_TGTTDR or I3C_TD(W)R for private read

Acknowledge? (by controller)

Acknowledge? (by controller)

Receiving and acknowledging a private read/write on the I3C bus

No transfer error?

Setting ERRF = 1

Transfer error

Setting I3C_EVR.FCF=1

Private read/write completed

On write: Receiving write data byte(s) into I3C_RD(W)R if any

On read: Transmitting read data byte(s) from I3C_TD(W)R

Update (or not) the configuration of a private transfer or the DEFGRPA/DEFTGTS CCC transfer:

write I3C_CFGR fields:

Configure TX-FIFO byte/word threshold: TXTHRES

Configure RX-FIFO byte/word threshold: RXTHRES

Enable/disable DMA mode for TX-FIFO: TXDMAEN

Enable/disable DMA mode for RX-FIFO: RXDMAEN

Updating status register I3C_SR

Read status: read I3C_SR (especially on private read or DEFTGTS/DEFGRPA CCC)

If controller early ended the private read (i.e. if I3C_SR.XDCNT < I3C_TGTTDR.TGTTDCNT):

flush TX-FIFO (i.e. write I3C_CFGR.TXFLUSH=1)

Updating status register I3C_SR

If I3C_DEVR0.DAVAL=1 and bus address = I3C_DEVR0.DA[6:0]

N

Y

If I3C_DEVR0.CREN=1 and I3C_DEVR0.DAVAL=1

N

Y

Enable I3C: write and set I3C_CFGR.EN=1

Configure (the execution mode of) a transfer via I3C_CFGR:

Configure TX-FIFO byte/word threshold: TXTHRES

Configure RX-FIFO byte/word threshold: RXTHRES

Enable/disable DMA mode for FIFOs: TXDMAEN, RXDMAEN

Enable S-FIFO for I3C_SR: SMODE=0

Configure interrupt/polling mode for any event: write I3C_IER

Initialize I3C as target (keeping I3C_CFGR.EN = 0):

Set I3C peripheral as target: write and clear I3C_CFGR.CRINIT=0

Set I3C bus available/idle/hand-off condition timings: write I3C_TIMINGR1.AVAL[7:0]

Enable/disable IBI request: write I3C_BCR, I3C_DCR, I3C_MAXRLR, I3C_MAXWLR, I3C_GETCAPR, I3C_CRCAP, I3C_GETMXDSR, I3C_EPIDR

If I3C_DEVR0.HJEN=1

If I3C_DEVR0.IBIEN=1 and I3C_DEVR0.DAVAL=1

N

Y

N

Y

N

Y

N

Y

N

Y
Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register.
- Set I3C bus timings via I3C timing register 1 (I3C_TIMINGR1): write AVAL[7:0] for:
  - Bus available condition time (tAVAL) for IBI or controller-role request
  - Bus idle condition time (tIDLE) for hot-join request
  - tNEWCRLock for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write I3C own device characteristics register (I3C_DEVR0) with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - I3C bus characteristics register (I3C_BCR)
  - I3C device characteristics register (I3C_DCR)
  - I3C maximum read length register (I3C_MAXRLR)
  - I3C maximum write length register (I3C_MAXWLR)
  - I3C get capability register (I3C_GETCAPR)
  - I3C controller-role capability register (I3C_CRCAPR)
  - I3C get max data speed register (I3C_GETMXDSR)
  - I3C extended provisioned ID register (I3C_EPIDR)
- Configure the execution mode of a transfer: I3C configuration register (I3C_CFGR), with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: I3C interrupt enable register (I3C_IER)

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message from the controller

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.
After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in Section 52.9:

- Figure 716: I3C CCC messages, as target
- Figure 717: I3C broadcast ENTDAA CCC, as target
- Figure 718: I3C broadcast DEFTGTS CCC, as target
- Figure 719: I3C broadcast DEFGRPA CCC, as target
- Figure 721: I3C private read/write messages, as target

Figure 712 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in Section 52.10.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating the I3C status register (I3C_SR), which can be read by the software after being notified by the corresponding flag in the I3C event register (I3C_EVR), or by the corresponding interrupt if enabled in the I3C interrupt enable register (I3C_IER).

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in Section 52.9:

- Figure 723: IBI transfer, as controller/target
- Figure 724: Hot-join request transfer, as controller/target
- Figure 725: Controller-role request transfer, as controller/target
Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via **I3C interrupt enable register (I3C_IER)**
- Modify following fields of the **I3C configuration register (I3C_CFGR)**:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the **I3C IBI payload data register (I3C_IBIDR)**, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write **I3C message control register [alternate] (I3C_CR)** with MTYPE[3:0] = 1010)
- Modify/prepare **I3C target transmit configuration register (I3C_TGTTDR)**, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC) from the controller

The registers usage vs. the I3C peripheral role as target is summarized in Section 52.8.1.

The static/dynamic registers fields usage, when acting as target, is summarized in Table 546.

### 52.8 I3C registers and programming

#### 52.8.1 I3C register set, as controller/target

**Table 544** lists the registers and their usage versus the I3C peripheral role.

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Used as target</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_CFGR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TGTTDR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_SR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_SER</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_RMR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_EVR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_IER</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_DEVR0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
### 52.8.2 I3C registers and fields use vs. peripheral state, as controller

When the I3C peripheral acts as controller, Table 545 lists the registers and their usage versus the controller state (disabled, idle, and active).

#### Table 545. I3C registers/fields usage versus controller state

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>

---

#### Table 544. I3C register usage (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Used as target</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_DEVRx x = 1..4</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_GETMXDSR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>I3C_EPIDR</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Register</td>
<td>Used as controller</td>
<td>Writable only in disabled state</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>I3C_CFRG</td>
<td>X</td>
<td>CRINIT HKSDAEN EN(1)</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TGWR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TGTTDR</td>
<td></td>
<td>-</td>
</tr>
<tr>
<td>I3C_SR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_SER</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>I3C_RMR</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>
Table 545. I3C registers/fields usage versus controller state (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as controller</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_EVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read (controller-role fields): HJF, CRF, IBIF, FCF, ERF, RXTGTENDF, RXFNEF, TXFNF, SFNEF, CFNF, RXLASTF, TXLASTF, TXFEF, CFEF</td>
</tr>
<tr>
<td>I3C_IER</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write xIE with x = HJ, CR, IB, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write (controller-role fields, refer to I3C_EVR)</td>
</tr>
<tr>
<td>I3C_DEV0</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write DA<a href="2">6:0</a></td>
</tr>
<tr>
<td>I3C_DEVRx</td>
<td>1..4</td>
<td>Write SUSP, IBIDEN, IBIACT, CRACK, DA<a href="2">6:0</a></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write(2)</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_GETMXDSR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_EPIDR</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).
52.8.3 I3C registers and fields usage vs. peripheral state, as target

When the peripheral acts as target, Table 546 lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 546. I3C registers/fields usage versus target state

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as target</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_CR</td>
<td>X</td>
<td></td>
<td>MTYPE[3:0] DCNT[2:0]</td>
<td></td>
</tr>
<tr>
<td>I3C_CFG0</td>
<td>X</td>
<td>CRINIT</td>
<td>Write: any used field except CRINIT, namely: TXDMAEN RXDMAEN TXTHRES RXTHRES TXFLUSH RXFLUSH</td>
<td></td>
</tr>
<tr>
<td>I3C_RDR</td>
<td>X</td>
<td></td>
<td></td>
<td>Read</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td>X</td>
<td></td>
<td></td>
<td>Read</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td>X</td>
<td></td>
<td>Write</td>
<td></td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td>X</td>
<td></td>
<td>Write</td>
<td></td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td>X</td>
<td></td>
<td>Write</td>
<td></td>
</tr>
<tr>
<td>I3C_TGTTDR</td>
<td>X</td>
<td></td>
<td>Write</td>
<td></td>
</tr>
<tr>
<td>I3C_SR</td>
<td>X</td>
<td></td>
<td>Read DIR, XDCNT[15:0]</td>
<td></td>
</tr>
<tr>
<td>I3C_SER</td>
<td>X</td>
<td></td>
<td>Read DOVR, STALL, PERR, CODERR[3:0]</td>
<td></td>
</tr>
<tr>
<td>I3C_RMR</td>
<td>X</td>
<td></td>
<td>Read RCODE[7:0]</td>
<td></td>
</tr>
</tbody>
</table>
### Table 5.46. I3C registers/fields usage versus target state (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as target</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_EVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Read (target-role fields): GRPF, DEFF, INTUPDF, ASUPDF, RSTF, MRLUPDF, MWLUPDF, DAUPDF, STAF, GETF, WKPF, CRUPDF, IBIENDF, ERRF, FCF, RXFNEF, TXFNF, TXLASTF, TXFEF</td>
</tr>
<tr>
<td>I3C_IER</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write (x)IE with (x = )GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td>Write (target-role fields, refer to I3C_EVR)</td>
</tr>
<tr>
<td>I3C_DEVR0</td>
<td>X</td>
<td>HJEN, CREN, IBIEN</td>
<td>Read RSTVAL, RSTACT[1:0], and AS[1:0]</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DEVRx</td>
<td>X = 1..4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXRLR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td></td>
<td>AVAL[7:0]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_CRCAPR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
I3C bus transfers and programming

52.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the peripheral acting as controller or target, is specified in Table 547.

---

Table 546. I3C registers/fields usage versus target state (continued)

<table>
<thead>
<tr>
<th>Register</th>
<th>Used as target</th>
<th>Writable only in disabled state</th>
<th>Typically written/read in idle state</th>
<th>Typically written/read in idle or active state</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C_GETMXDSR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I3C_EPIDR</td>
<td>X</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.

2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.
## Table 547. List of supported I3C CCCs, as controller/target

<table>
<thead>
<tr>
<th>CCC name</th>
<th>CCC value</th>
<th>Read/write</th>
<th>With/without defining byte</th>
<th>With/without optional data byte(s)</th>
<th>Use as controller</th>
<th>Use as target, raised I3C_EVR event</th>
<th>When target: specific action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Broadcast CCCs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENEC</td>
<td>0x00</td>
<td>No defining/sub-command byte</td>
<td>With one data byte (enable target events byte)</td>
<td>X X, INTUPDF</td>
<td>Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISEC</td>
<td>0x01</td>
<td>No defining/sub-command byte</td>
<td>With one data byte (disable target events byte)</td>
<td>X X, INTUPDF</td>
<td>Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ENTASx</strong></td>
<td>x = 0...3</td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X X, ASUPDF</td>
<td>Update I3C_DEVR0.AS[1:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTDAA</td>
<td>0x06</td>
<td>No defining/sub-command byte</td>
<td>-</td>
<td>X X, DAUPDF</td>
<td>Clear I3C_DEVR0.DAVAL = 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>ENTDAA</strong></td>
<td>0x07</td>
<td>No defining/sub-command byte</td>
<td>-</td>
<td>X X, DAUPDF</td>
<td>Update I3C_DEVR0: DA[6:0] and set DAVAL = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DEFTGTS</strong></td>
<td>0x08</td>
<td>Write</td>
<td>With [1+ 4x (1+ number_of_targets)] x data bytes</td>
<td>X X, DEFF</td>
<td>Update I3C_RDR/ I3C_RDWR. Refer to Figure 718.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETMWL</td>
<td>0x09</td>
<td>No defining/sub-command byte</td>
<td>With two data byte</td>
<td>X X, MWLUPDF</td>
<td>Update I3C_MAXWLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETMRL</td>
<td>0x0A</td>
<td>No defining/sub-command byte</td>
<td>With 2 or 3 data bytes</td>
<td>X X, MRLUPDF</td>
<td>Update I3C_MAXRLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENTTM</td>
<td>0x0B</td>
<td>No defining/sub-command byte</td>
<td>With one data byte</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETXTIME</td>
<td>0x28</td>
<td>With sub-command byte</td>
<td>Without or with one or more data bytes</td>
<td>X X</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETAASA</td>
<td>0x29</td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X X, RSTF after detected reset pattern</td>
<td>Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTACT</td>
<td>0x2A</td>
<td>With defining byte (0x00, 0x01 or 0x02)</td>
<td>No data byte</td>
<td>X X, GRPF</td>
<td>Update I3C_RDR/ RDWR. Refer to Figure 719.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DEFGRPA</strong></td>
<td>0x2B</td>
<td>No defining/sub-command byte</td>
<td>With several data bytes</td>
<td>X X, GRPF</td>
<td>Update I3C_RDR/ RDWR. Refer to Figure 719.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTGRPA</td>
<td>0x2C</td>
<td>No defining/sub-command byte</td>
<td>No data byte</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Table 547. List of supported I3C CCCs, as controller/target (continued)

<table>
<thead>
<tr>
<th>CCC name</th>
<th>CCC value</th>
<th>Read /write</th>
<th>With/without defining byte</th>
<th>With/without optional data byte(s)</th>
<th>Use as controller</th>
<th>Use as target, raised I3C_EVR event</th>
<th>When target: specific action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Direct CCCs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENEC</td>
<td>0x80</td>
<td>No defining/sub-command byte</td>
<td>With one data byte (enable target events byte)</td>
<td>X, X, INTUPDF</td>
<td>Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISEC</td>
<td>0x81</td>
<td>Write</td>
<td>No defining/sub-command byte</td>
<td>With one data byte (disable target events byte)</td>
<td>X, X, INTUPDF</td>
<td>Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any</td>
<td></td>
</tr>
<tr>
<td>ENTASx</td>
<td>0x82..0x85</td>
<td>x = 0...3</td>
<td></td>
<td>No data byte</td>
<td>X, X, ASUPDF</td>
<td>Update I3C_DEVR0.AS[1:0]</td>
<td></td>
</tr>
<tr>
<td>SETDASA</td>
<td>0x87</td>
<td>Write</td>
<td>No data byte</td>
<td>X</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SETNEWDA</td>
<td>0x88</td>
<td>With one data byte</td>
<td>With two data bytes</td>
<td>X, DAUPDF</td>
<td>Update I3C_DEVR0: DA[6:0] (and set DAVAL = 1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETMWL</td>
<td>0x89</td>
<td>With two data bytes</td>
<td>With two or three data bytes</td>
<td>X, MWLUPDF</td>
<td>Update I3C_MAXWLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SETMRL</td>
<td>0x8A</td>
<td>With two or three data bytes</td>
<td></td>
<td>X, MRLUPDF</td>
<td>Update I3C_MAXRLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GETMWL</td>
<td>0x8B</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>With two data bytes</td>
<td>X, X, GETF</td>
<td>Return data bytes from I3C_MAXWLR[15:0]. Refer to Section 52.16.19.</td>
<td></td>
</tr>
<tr>
<td>GETMRL</td>
<td>0x8C</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>With two or three data bytes</td>
<td>X, X, GETF</td>
<td>Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to Section 52.16.18.</td>
<td></td>
</tr>
<tr>
<td>CCC name</td>
<td>CCC value</td>
<td>Read/write</td>
<td>With/without defining byte</td>
<td>With/without optional data byte(s)</td>
<td>Use as controller</td>
<td>Use as target, raised I3C_EV event</td>
<td>When target: specific action</td>
</tr>
<tr>
<td>-----------</td>
<td>-----------</td>
<td>------------</td>
<td>-----------------------------</td>
<td>-----------------------------------</td>
<td>-------------------</td>
<td>-----------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>GETPID</td>
<td>0x8D</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>With six data bytes</td>
<td>X, GETF</td>
<td>Return data bytes from I3C_EPIDR. Refer to Section 52.16.28.</td>
<td></td>
</tr>
<tr>
<td>GETBCR</td>
<td>0x8E</td>
<td>Write</td>
<td>No defining/sub-command byte</td>
<td>With one data byte</td>
<td>X, GETF</td>
<td>Return data byte from I3C_BCR[7:0]. Refer to Section 52.16.23.</td>
<td></td>
</tr>
<tr>
<td>GETDCR</td>
<td>0x8F</td>
<td>Write</td>
<td>With or without defining byte (TGTSTAT, PRECR)</td>
<td>With two data bytes (format 1 or format 2 with PRECR)</td>
<td>X</td>
<td>Return I3C_DCR[7:0]. Refer to Section 52.16.24.</td>
<td></td>
</tr>
<tr>
<td>GETSTATUS</td>
<td>0x90</td>
<td>Read</td>
<td>With or without defining byte (TGTSTAT, PRECR)</td>
<td>With two data bytes (format 1 or format 2 with PRECR)</td>
<td>X, STAF if format 1 X, GETF if format 2</td>
<td>Return 2 data bytes, as detailed in Section 52.9.9.</td>
<td></td>
</tr>
<tr>
<td>GETACCCR</td>
<td>0x91</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>With one data byte</td>
<td>X, CRUPDF</td>
<td>Return data byte from I3C_DEVR0.DA[6:0] with parity bit</td>
<td></td>
</tr>
<tr>
<td>GETMXDS</td>
<td>0x94</td>
<td>Write</td>
<td>With or without defining byte (WRRDTURN, CRHDLY)</td>
<td>With two data bytes (format 1 or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)</td>
<td>X, GETF</td>
<td>Return data byte(s) from I3C_GETMXDSR. Refer to Section 52.16.27.</td>
<td></td>
</tr>
<tr>
<td>GETCAPS</td>
<td>0x95</td>
<td>Write</td>
<td>With or without defining byte (TGTSTAT, CRCAPS)</td>
<td>With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)</td>
<td>X, GETF</td>
<td>Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to Section 52.16.25) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to Section 52.16.26)</td>
<td></td>
</tr>
<tr>
<td>D2DXFER</td>
<td>0x97</td>
<td>Write</td>
<td>With defining byte</td>
<td>With defining byte</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SETXTIME</td>
<td>0x98</td>
<td>Write</td>
<td>With sub-command byte</td>
<td>With sub-command byte</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GETXTIME</td>
<td>0x99</td>
<td>Read</td>
<td>No defining/sub-command byte</td>
<td>No defining/sub-command byte</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
52.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

*Figure 713* illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

<table>
<thead>
<tr>
<th>CCC name</th>
<th>CCC value</th>
<th>Read/write</th>
<th>With/without defining byte</th>
<th>With/without optional data byte(s)</th>
<th>Use as controller</th>
<th>Use as target, raised I3C_EVR event</th>
<th>When target: specific action</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSTACT</td>
<td>0x9A</td>
<td>Read/Write</td>
<td>With defining byte (0x00, 0x01, or 0x02)</td>
<td>With defining byte (0x00, 0x01, or 0x02)</td>
<td>X</td>
<td>X, RSTF if detected reset pattern</td>
<td>Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1</td>
</tr>
<tr>
<td>SETGRPA</td>
<td>0x9B</td>
<td>Write</td>
<td>No defining/sub-command byte</td>
<td>No defining/sub-command byte</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>RSTGRPA</td>
<td>0x9C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 547. List of supported I3C CCCs, as controller/target (continued)
Figure 713. I3C CCC messages, as controller

**LEGEND**

- **Controller**
  - (driver) SDA low/high 2 in open-drain (arbitrable header)
  - SDA is open-drain
- **Target**
  - SDA is push-pull
- Acknowledge from the addressed target(s) (which drives SDA low in open-drain)
- Acknowledge from the addressed target(s) (which drives SDA low in open-drain) without hand-off (ie continuous driving SDA)
- Sr is followed by 7'h7E+W broadcast address at the end of a direct CCC message
- Sr*: Repeat for each addressed target
- Transition bit (parity bit for write data) from controller (drives SDA in push-pull)
- Transition bit (end of data for read data) from controller and/or from target (drives SDA low/high-Z in open-drain)

---

**RM0477**

**Improved inter-integrated circuit (I3C)**

**MSv69015V3**
52.9.3 **I3C broadcast ENTDAA CCC transfer, as controller**

*Figure 714* illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

52.9.4 **I3C broadcast/direct RSTACT CCC transfer, as controller**

*Figure 715* illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.
Figure 715. I3C broadcast, direct read and direct write RSTACT CCC, as controller

**LEGEND**

- Controller and target (drives) SDA low/high Z in open-chain (arbitrable header)
- Controller drives SDA in push-pull
- Target drives SDA in push-pull
- Acknowledge from the addressed target(s) (which drives SDA low in open-drain)
- Acknowledge without hand-off (which drives SDA low in open-drain)

**Controller (and target) drives SDA low/high Z in open-chain (arbitrable header)**

**Controller drives SDA in push-pull**

**Target drives SDA in push-pull**

**Acknowledgment from the addressed target(s) (which drives SDA low in open-drain)**

**Acknowledgment without hand-off (which drives SDA low in open-drain)**

---

**I3C broadcast CCC write**

**I3C direct CCC write**, first part

**I3C direct CCC write**, second part

**I3C direct CCC read**, first part

**I3C direct CCC read**, second part
52.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGPRA), as target

*Figure 716* illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGPRA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.
Figure 716. I3C CCC messages, as target

**I3C broadcast CCC write**
-(except ENTDAA, DEFTGTS, DEFGRPA)
(when IP acts as target)

**I3C direct CCC write**
-(when IP acts as target)

**I3C direct CCC read**
-(when IP acts as target)

---

**LEGEND**

- Controller (and target) drive(s) SDA low/high-Z in open-drain (arbitrable header)
- Controller drives SDA in open-drain (and SCL in open-drain for I2C vs in push-pull for I3C)
- Sr(*) is followed by 7'h7E+W broadcast address at the end of a direct CCC message
- Transition bit (parity bit for write data) from controller (drives SDA in push-pull)
- Transition bit (end of data for read data) from controller and/or from target (drives SDA high-Z in open-drain)
52.9.6 I3C broadcast ENTDAA CCC transfer, as target

*Figure 717* illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

**Figure 717. I3C broadcast ENTDAA CCC, as target**

---

**LEGEND**

- Controller (or target) drive(s) SDA low/high Z in open-drain
- Controller drives SDA in open-drain (start or stop condition)
- Controller drives SDA in push-pull
- Target(s) drive(s) SDA low/high Z in open-drain (and controller in open-drain high Z)
- Acknowledge from the addressed target(s) (which drive(s) SDA low in open-drain) with hand-off (ie switch back SDA control to controller)
- Acknowledge from the addressed target(s) (which drive(s) SDA low in open-drain) without hand-off (ie continues driving SDA low/high Z in open-drain)
- Transition bit (parity bit for CCC) from controller (drives SDA in push-pull)

48-bit unique/provisioned ID is subject to arbitration. The target which wins arbitration continues with providing its BCR and DCR.

(*)
52.9.7 I3C broadcast DEFTGTS CCC transfer, as target

*Figure 718* illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

*Figure 718. I3C broadcast DEFTGTS CCC, as target*

---

**LEGEND**

- Controller (and target) drive(s) SDA low/high Z in open-drain (arbitrable header)
- Controller drives SDA in open-drain
- Controller drives SDA in push-pull
- Acknowledge from the addressed target(s) (which drive(s) SDA low in open-drain) with hand-off (i.e. switch back SDA control to controller)
- Transition bit (parity bit for CCC) from controller (drives SDA in push-pull)
52.9.8  I3C broadcast DEFGRPA CCC transfer, as target

Figure 719 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 719. I3C broadcast DEFGRPA CCC, as target

52.9.9  I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
• STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0
The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:
• STATUS[15:8] = 0000 0000 (unused)
• STATUS[7:2] = 0000 (unused)
• STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
• STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).
Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).
### 52.9.10 I3C private read/write transfer, as controller

*Figure 720 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.*

*Figure 720. I3C private read/write messages, as controller*

### 52.9.11 I3C private read/write transfer, as target

*Figure 721 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.*
Figure 721. I3C private read/write messages, as target

**LEGEND**

- Controller (and target) drive(s) SDA low/high Z in open-drain (arbitrable header)
- Controller drives SDA in open-drain
- Controller drives SDA in push-pull
- Target drives SDA in push-pull
- Transition bit (parity bit for write data) from controller (drives SDA in push-pull)
- Transition bit (end of data for read data) from controller and/or from target (drive SDA low/high Z in open-drain)

(*) Target is not able to acknowledge an addressed private read/write if it is emitted without arbitrable header (i.e. after a start) and if it occurs at the same SCL clock cycle as an IBI request from this target. In order to avoid such an address-based arbitration collision, the controller should emit a private read/write with arbitrable header when the IBI request is enabled.
52.9.12 Legacy I²C read/write transfer, as controller

Figure 722 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

Figure 722. Legacy I²C read/write messages, as controller
52.9.13 I3C IBI transfer, as controller/target

*Figure 723* illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

*Figure 723. IBI transfer, as controller/target*
When the peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

### 52.9.14 I3C hot-join request transfer, as controller/target

Figure 724 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

![Figure 724. Hot-join request transfer, as controller/target](MSv69019V2.png)

**LEGEND**

- Controller drives SDA in push-pull
- Target drives SDA in open-drain (and controller in open-drain highZ)
- Target initiates a start by driving SDA low in open-drain while SCL is high and waits for controller to pull SCL low


demote **("**) Repeated start (Sr) if there is a pending transfer to execute, else stop (P)

```markdown
I3C hot-join request from target i
(when IP is acting as controller)

Start request

Bus idle condition

Reserved Address

I3C hot-join request from target j
(when IP is acting as controller)

Start request

Bus free condition (managed by target)

Reserved Address

Bus free condition (managed by controller)

Reserved Address

ACK

Sr or P (**)

Sr or P (**)

```
52.9.15  I3C controller-role request transfer, as controller/target

Figure 725 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 725. Controller-role request transfer, as controller/target

---

**Legend**

- **Controller drives SDA in push-pull**
  - Controller drives SDA in push-pull (start condition)
  - Controller drives SDA in push-pull (stop condition)
- **Target drives SDA low/highZ in open-drain**
  - Target drives SDA low/highZ in open-drain (start condition)
  - Target drives SDA low/highZ in open-drain (stop condition)
- **S**
  - Controller provides SCL clocking after t_{SCL}
- **CAS**
  - Target initiates a start by driving SDA low in open-drain while SCL is high and waits for controller to pull SCL low
- **Sr or P**
  - Repeated start (Sr) if there is a pending transfer to execute, else stop (P)

---

(1): Target is not able to acknowledge an addressed private read/write if it is entitled without arbitrable header (i.e., after a start) and if it occurs at the same SCL clock cycle as an IBI request from this target. In order to avoid such an address-based arbitration collision, the controller should emit a private read/write with arbitrable header when the IBI request is enabled.

(2): Repeated start (Sr) if there is a pending transfer to execute, else stop (P)
52.10 I3C FIFOs management, as controller

52.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of Section 52.9, C-FIFO can be used during any of the following transfers:

- broadcast CCC (Figure 713, Figure 714, Figure 715)
- direct read/write CCC (Figure 715)
  - command part, first message
  - data part, next message(s)
- private read/write (Figure 720)
- legacy I2C read/write (Figure 722)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 726 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

![Figure 726. C-FIFO management, as controller](image)

1. This feature is implementation-dependent and can be unavailable. Refer to Section 52.3.4.
First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

### 52.10.2 TX-FIFO management, as controller

When controller, as shown in figures of Section 52.9, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (Figure 713, Figure 714, Figure 715)
- private write (Figure 720)
- legacy I2C write (Figure 722)

*Figure 727* illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.
First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- **TXDMAEN**: enable/disable DMA mode for TX-FIFO
- **TXTHRES**: push-on data byte(s) or word(s) into TX-FIFO
- **TMODE**: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CFGR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CFGR register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.
An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in Table 539, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

- wait for a first control word to be written into the C-FIFO
- wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size
- If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
  - Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
  - If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control
word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

52.10.3 RX-FIFO management, as controller

When controller, as shown in figures of Section 52.9, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (Figure 714)
- direct CCC read (Figure 713), including direct RSTACT CCC read (Figure 715)
- private read (Figure 720)
- legacy I2C read (Figure 722)

Figure 728 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

**Figure 728. RX-FIFO management, as controller**
First, the software must initialize the RX-FIFO management via the following fields of the 
I3C_CGFR register:
• RXDMAEN: enable/disable DMA mode for RX-FIFO
• RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:
• directly by the software (RXDMAEN = 0) at the byte/word level:
  – via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data 
    byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) 
    before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit 
    RXTHRES in the I3C_CGFR register
  – via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
• by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request 
  from the I3C peripheral (i3c_rx_dma):
  – as configured at DMA block level, the DMA automatically pops-out-reads data 
    bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit 
    RXTHRES), and writes them into its memory destination buffer, until the frame 
    completion (a stop is emitted on the I3C bus after the last message of the frame), 
    unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated 
start. At message level, the last received data byte/word from the I3C bus is flagged by 
IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple 
messages (separated by a repeated start), this event can be used by the software for 
updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next 
message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is 
reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If 
enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is 
not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the 
addressed target.
• If RXDMAEN = 1:
  – the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, 
    capable of peripheral early termination (refer to DMA implementation section).
  – The software must configure the DMA channel x to enable the DMA peripheral-
    flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to 
    perform a(n) (early or not) DMA block completion.
  – Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx 
    and/or DMA_CxSAR register(s), to get the effective number of DMA transferred 
    bytes.
• If RXDMAEN = 0:
  – at message level, the last received data byte/word from the I3C bus is flagged by 
    RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.
In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message, and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to I3C status register (I3C_SR) and Section 52.10.4.

### 52.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the I3C status register (I3C_SR) for each transferred message.

#### Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferred a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to Section 52.10.3).

#### Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. Figure 729 illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.
First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.
The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

52.11 I3C FIFOs management, as target

52.11.1 RX-FIFO management, as target

When target, as shown in figures of Section 52.9, the RX-FIFO is used during any of the following received and acknowledged transfers:
- broadcast DEFTGTS CCC (Figure 718)
- broadcast DEFGRPA CCC (Figure 719)
- private write (Figure 721)

Figure 730 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as target.

Figure 730. RX-FIFO management, as target
First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CF CF / CDEFF / CGRP F to 1 in the I3C_CEV R register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

52.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of Section 52.9, the TX-FIFO is used only during a private read (Figure 721).

Figure 731 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.
First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus, the software must configure the I3C target transmit configuration register (I3C_TGTTDR) to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the I3C bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

**Note:** TX-FIFO size is 8 bytes (refer to Table 539).
Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- **directly by the software (TXDMAEN = 0) at the byte/word level:**
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGFR register
  - via enabled interrupt notification if TXFNFIE = 1

- **by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):**
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the I3C bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the I3C bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the I3C bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGFR register, continue (or not) for another private read, depending upon the user application

If TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

**Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size**

Alternatively to the use of the I3C_TGTTDR register, as shown in the Figure 732, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the I3C bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.
52.12 I3C error management

52.12.1 Controller error management

Table 548 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Controller action</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE0</td>
<td>Illegally formatted CCC (e.g. less returned data byte(s))</td>
<td>Prematurely ended read data by target on a direct CCC read(2)</td>
<td>Hardware emits a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000</td>
</tr>
</tbody>
</table>
### Table 548. I3C controller error management (continued)

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Controller action</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE1</td>
<td>Monitoring error</td>
<td>An incorrect ACK is detected at the end of a legacy I2C read</td>
<td>Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Restart or stop cannot be generated at the end of an I3C SDR read</td>
<td>SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>After a CE1 error, a start cannot be generated</td>
<td>Hardware emits an HDR exit pattern followed by a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010</td>
</tr>
<tr>
<td>CE2</td>
<td>No response to broadcast address (0b111_1110)</td>
<td>Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message</td>
<td>Hardware emits an HDR exit pattern followed by a stop.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010</td>
</tr>
<tr>
<td>CE3</td>
<td>Failed controller hand-off</td>
<td>New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed</td>
<td>Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.</td>
<td>ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC</td>
<td>Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.</td>
<td>ERRF = 1 and DERR = 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Addressed target is NACK-ed on a direct CCC read for the first time</td>
<td>Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I2C, or a direct CCC read for the second time</td>
<td>Hardware emits a stop.</td>
<td>ERRF = 1 and ANACK= 1</td>
</tr>
</tbody>
</table>
Table 548. I3C controller error management (continued)

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Controller action</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Assigned/transmitted dynamic address with parity bit is NACK-ed on an ENTDAA CCC for the first time</td>
<td>NACK is detected</td>
<td>Hardware performs a single-retry assignation loop by emitting a Restart + 0b111 1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>Assigned/transmitted dynamic address with parity bit is NACK-ed on an ENTDAA CCC for the second time</td>
<td>NACK is detected</td>
<td>Hardware emits a stop.</td>
<td>ERF = 1 and DNACK= 1</td>
</tr>
<tr>
<td>-</td>
<td>Write data is NACK-ed on a legacy I2C write</td>
<td>SCL stall timeout</td>
<td>Hardware emits a stop.</td>
<td>ERF = 1 and (COVR = 1 or DOVR = 1)</td>
</tr>
<tr>
<td>-</td>
<td>Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. ERF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.

2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4. On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

52.12.2 Target error management

Table 549 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 549. I3C target error management

<table>
<thead>
<tr>
<th>Error type</th>
<th>Description</th>
<th>Error detection</th>
<th>Next (when emitted by the controller)</th>
<th>Reported error(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TE0</td>
<td>Invalid broadcast address (0b111 1110+RnW = 0) or</td>
<td>A forbidden address is detected after a start or a repeated start</td>
<td>Hardware waits for an HDR exit pattern</td>
<td>ERF = 1 and PERR = 1 and ODERR[3:0] = 1000</td>
</tr>
<tr>
<td></td>
<td>Invalid 7-bit dynamic address + RnW = 1 after DAA assignment</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TE1</td>
<td>CCC code</td>
<td>A CCC code is detected with a parity error</td>
<td></td>
<td>ERF = 1 and PERR = 1 and ODERR[3:0] = 1001</td>
</tr>
</tbody>
</table>
### 52.13 I3C wake-up from low-power mode(s)

#### 52.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section 7: Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.
When entering a Stop mode, the V\textsubscript{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

**As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request**

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTAS\textsubscript{x} CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 $\mu$s, 100 $\mu$s, 2 ms or 50 ms, has elapsed. This delay defines the T\textsubscript{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTAS\textsubscript{x} with the value x=1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T\textsubscript{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.

2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.

3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 723*), and it maintains the APB clock request until that IBIF is cleared.
     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 724*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 725*), and it maintains the APB clock request until that HJF is cleared.

4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awakened.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.
7. With the APB clock running, the peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

**As target: wake-up on a reset pattern**
The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:
1. When the peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register.
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start
1. The peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high)
   - As target, as initiated by an external I3C device, whatever controller or target.
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a I3C bus start is missed and waits for the kernel clock to be provided. It can be noted that an I3C controller message intended to be addressed to it may have been missed (and NAcked). Or may have been missed any I3C CCC or private message from the controller to another addressed target or an IBI/CR/HJF start request from another target.
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs TCAS latency).
3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   - The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that an I3C bus transaction was missed, but not whether the target was addressed or not. The software should use a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.
52.14  I3C in low-power modes

Table 550. Effect of low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. I3C interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(^{(1)})</td>
<td>The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode. I3C transfers can occur and can be assisted with an autonomous DMA for data transfers to/from memory, provided that the autonomous DMA can operate in Stop mode. Refer to Section 52.13 for more details.</td>
</tr>
<tr>
<td>Standby</td>
<td>The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

\(^{(1)}\) Refer to Table 538 to know if Stop mode is supported, and which one.
### I3C interrupts

#### Table 551. I3C interrupt requests

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Used as Interrupt enable: field in I3C_IER</th>
<th>Event flag: field in I3C_EVR</th>
<th>Event clear method: write 1 to field in I3C_CEVR</th>
</tr>
</thead>
<tbody>
<tr>
<td>I3C EVT</td>
<td>A control word is requested</td>
<td>X -</td>
<td>CFNFIE</td>
<td>CFNFF</td>
</tr>
<tr>
<td>I3C</td>
<td>A status word is available</td>
<td>X -</td>
<td>SFNEIE</td>
<td>SFNEF</td>
</tr>
<tr>
<td>I3C</td>
<td>Data to be transmitted are requested</td>
<td>X X</td>
<td>TXFNIE</td>
<td>TXFNFF</td>
</tr>
<tr>
<td>I3C</td>
<td>Received data are available</td>
<td>X X</td>
<td>RXFNEIE</td>
<td>RXFNFF</td>
</tr>
<tr>
<td>I3C</td>
<td>Controller: frame transfer is completed Target: private transfer is completed</td>
<td>X X</td>
<td>FCIE</td>
<td>FCF</td>
</tr>
<tr>
<td>I3C</td>
<td>A private read transfer is prematurely ended by the target (and SMODE = 0 in the I3C_CFG register)</td>
<td>X -</td>
<td>RXTGTENDIE</td>
<td>RXTGTENDF</td>
</tr>
<tr>
<td>I3C</td>
<td>An IBI request is received</td>
<td>X -</td>
<td>IBIIE</td>
<td>IBIF</td>
</tr>
<tr>
<td>I3C</td>
<td>A controller-role request is received</td>
<td>X -</td>
<td>CRIE</td>
<td>CRF</td>
</tr>
<tr>
<td>I3C</td>
<td>A hot-join request is received</td>
<td>X -</td>
<td>HJIE</td>
<td>HJF</td>
</tr>
<tr>
<td>I3C</td>
<td>IBI request is completed</td>
<td>- X</td>
<td>IBIENDIE</td>
<td>IBIENDF</td>
</tr>
<tr>
<td>I3C</td>
<td>I3C bus start is missed</td>
<td>- X</td>
<td>WKPIE</td>
<td>WKPF</td>
</tr>
<tr>
<td>I3C</td>
<td>Direct GETACCR CCC is received</td>
<td>- X</td>
<td>CRUPDIE</td>
<td>CRUPDF</td>
</tr>
<tr>
<td>I3C</td>
<td>Direct GETSTATUS CCC is received</td>
<td>- X</td>
<td>STAIE</td>
<td>STAF</td>
</tr>
<tr>
<td>I3C</td>
<td>Any direct GETxxx CCC (except GETSTATUS) is received</td>
<td>- X</td>
<td>GETIE</td>
<td>GETF</td>
</tr>
<tr>
<td>I3C</td>
<td>Dynamic address is updated (broadcast ENTDAA or RSTDAA, or direct SETNEWDA is received)</td>
<td>- X</td>
<td>DAUPDIE</td>
<td>DAUPDF</td>
</tr>
<tr>
<td>I3C</td>
<td>Direct SETMWL CCC is received</td>
<td>- X</td>
<td>MWLUPDIE</td>
<td>MWLUPDF</td>
</tr>
<tr>
<td>I3C</td>
<td>Direct SETMRL CCC is received</td>
<td>- X</td>
<td>MRLUPDIE</td>
<td>MRLUPDF</td>
</tr>
<tr>
<td>I3C</td>
<td>A reset pattern is detected</td>
<td>- X</td>
<td>RSTIE</td>
<td>RSTF</td>
</tr>
<tr>
<td>I3C</td>
<td>Bus activity state is updated (direct/broadcast ENTASx CCC is received)</td>
<td>- X</td>
<td>ASUPDIE</td>
<td>ASUPDF</td>
</tr>
<tr>
<td>I3C</td>
<td>Broadcast/direct ENEC/DISEC CCC is received</td>
<td>- X</td>
<td>INTUPDIE</td>
<td>INTUPDF</td>
</tr>
<tr>
<td>I3C</td>
<td>Broadcast DEFSGTS CCC is received</td>
<td>- X</td>
<td>DEFIE</td>
<td>DEFF</td>
</tr>
<tr>
<td>I3C</td>
<td>Broadcast DEFGPRA CCC is received</td>
<td>- X</td>
<td>GRPIE</td>
<td>GRPF</td>
</tr>
<tr>
<td>ERR</td>
<td>An error occurred</td>
<td>X X</td>
<td>ERRIE</td>
<td>ERRF</td>
</tr>
</tbody>
</table>
52.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

52.16.1 I3C message control register (I3C_CR)

Address offset: 0x000
Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description Section 52.16.2 must be considered.

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
</tbody>
</table>

Bit 31 MEND: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted
1: this message from controller ends with a stop (P), being the last message of a frame
** improved inter-integrated circuit (I3C) **

** Bits 30:27 **

** MTYPE[3:0]: ** Message type (whatever I3C acts as controller/target)

** Condition: when I3C acts as I3C controller **

0000: SCL clock is forced to stop until a next control word is executed

  Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL "stuck at" recovery. Refer to Table 548.

0001: header message

  Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to Table 548 and MIPI specification about escalation handling.

0010: private message (refer to Figure 720)

  Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
  Bit[16] (RNW) is the emitted RnW bit.
  Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.

  The transferred private message is:
  - (S / S + 0b111_1110 + RnW = 0 + Sr/Sr* + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
  - After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
  - Sr*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: direct message (second part of an I3C SDR direct CCC command) (refer to Figure 713)

  Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
  Bit[16] (RNW) is the emitted RnW bit.
  Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.

  The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: legacy I2C message (refer to Figure 722)

  Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
  Bit[16] (RNW) is the emitted RnW bit.
  Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.

  The transferred legacy I2C message is:
  - (S / S + 0b111_1110 + RnW = 0 + Sr/Sr* + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
  - After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
  - Sr*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

** Condition: when I3C acts as I3C target **

1000: hot-join request (W) (refer to Figure 724)

  The transferred hot-join request is (S +) 0b000_0010 addr + RnW = 0.
  Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: controller-role request (W) (refer to Figure 725)

  The transferred controller-role request is (S +) DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: IBI (in-band interrupt) request (R) (refer to Figure 723)

  Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
  The transferred IBI request is (S +) DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message. When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed or the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

** Bits 26:24 **

Reserved, must be kept at reset value.
Improved inter-integrated circuit (I3C)  

2362/3791 RM0477 Rev 6

52.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000
Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to Section 52.16.1.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO, either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DCNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
<tr>
<td>w</td>
</tr>
</tbody>
</table>

Bits 23:17 ADD[6:0]: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 RNW: Read / non-write message (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.
0: write message
1: read message

Bits 15:0 DCNT[15:0]: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.
When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).
Linear encoding up to 64 Kbytes - 1
0x0000: no data to transfer
0x0001: 1 byte
0x0002: 2 bytes
... 0xFFFF: 64 Kbytes - 1 byte
Bit 31  **MEND**: Message end type/last message of a frame (when I3C acts as controller)  
0: this message from controller is followed by a repeated start (Sr), before another message must be emitted  
1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27  **MTYPE[3:0]**: Message type (when I3C acts as controller)  

**Condition: when I3C acts as I3C controller**

**0110**: broadcast/direct CCC command (refer to Table 547, Figure 713, Figure 714, Figure 715)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code  
Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.  
If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command  
The transferred direct CCC command (first part) message is:  
– {S / S + 0b111_1110 + RnW = 0 / Sr*} + (direct CCC + T) + (8-bit Data + T)* + Sr  
– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.  
– Sr*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to Figure 714)

The transferred broadcast CCC command message is:  
– {S / S + 0b111_1110 + RnW = 0 / Sr*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P  
– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.  
– Sr*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

**Others**: reserved

Bits 26:24  Reserved, must be kept at reset value.

Bits 23:16  **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)  
If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.  
If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0  **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes  
Linear encoding up to 64 Kbytes - 1.  
0x0000: no data to transfer.  
0x0001: 1 byte  

**Note**:  Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 714).  
0x0002: 2 bytes  

...  
0xFFFF: 64 Kbytes - 1 byte
52.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004
Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | TSFSET: Frame transfer set (software trigger) (when I3C acts as controller) |
|        | This bit can only be written. When I3C acts as I3C controller: |
|        | 0: no action |
|        | 1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed) |
|        | Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF. |
| Bits 29:24 | Reserved, must be kept at reset value. |
| Bit 23 | Reserved, must be kept at reset value. |
| Bit 22 | Reserved, must be kept at reset value. |
| Bit 21 | CFLUSH: C-FIFO flush (when I3C acts as controller) |
|        | This bit can only be written. |
|        | 0: no action |
|        | 1: flush C-FIFO |
Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:
0: DMA mode is disabled for C-FIFO
   - Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
   - A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).
1: DMA mode is enabled for C-FIFO
   - DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
   - A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.
0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.
   - A frame transfer starts as soon as the first control word is present in C-FIFO.
1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to Section 52.10.2 for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.
When I3C acts as target, this bit must be cleared.
0: S-FIFO is disabled
   - Status register (I3C_SR) is used without FIFO mechanism.
   - There is no SCL stalling if a new status register content is not read.
   - Status register must be read before being overwritten by the hardware.
   - Must have SDMAEN = 0 in the I3C_CFG register.
1: S-FIFO is enabled.
   - Each message status must be read.
   - There is SCL stalling when the S-FIFO is full and a next message status must be read.
   - S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.
0: no action
1: flush S-FIFO
Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFG register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR
- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).
1: DMA mode is enabled for reading status register I3C_SR
- Must have SMODE = 1 in the I3C_CFG register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 **Reserved, must be kept at reset value.**

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold
- TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).
1: 1-word / 4-byte threshold
- TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action
1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO
- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).
1: DMA mode is enabled for TX-FIFO
- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).
Bit 11  Reserved, must be kept at reset value.

Bit 10  **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)
This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).
0: 1-byte threshold
RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).
1: 1-word/4-bytes threshold
RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9  **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)
This bit can only be written.
0: no action
1: flush RX-FIFO

Bit 8  **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)
0: DMA mode is disabled for RX-FIFO
- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).
1: DMA mode is enabled for RX-FIFO
- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the assert/ed RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7  **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)
0: hot-join request is not acknowledged
After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).
1: hot-join request is acknowledged
After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6  Reserved, must be kept at reset value.

Bit 5  **HKSDAEN**: High-keeper enable on SDA line (when I3C acts as a controller)
0: High-keeper is disabled
1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

*Note: This bit can be modified only when EN = 0 in the I3C_CFG register.*
Improved inter-integrated circuit (I3C) RM0477

Bit 4 **EXITPTRN**: HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header (S/Sr + 0x7E addr + W) is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN**: HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to insure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the peripheral is under reset (partial reset).

- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.

- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)
52.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010
Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **RDB[7:0]**: 8-bit received data on I3C bus.

52.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014
Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received
word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVRF register when acting as controller.

- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVRF register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

### 52.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018  
Reset value: 0x0000 0000  

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVRF register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVRF register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:  
  - when PRELOAD = 1 in the I3C_TGTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non
full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.

- when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

### 52.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.

- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0]
in the I3C_TGTTDR register.

- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERF = 1 in the I3C_EVR register and a data underrun flag DOP = 1 in the I3C_SER register (and corresponding interrupt if enabled).

| Bits 31:24 | TDB3[7:0] | 8-bit transmit data (latest byte on I3C bus). |
| Bits 23:16 | TDB2[7:0] | 8-bit transmit data (next byte after TDB1[7:0] on I3C bus). |
| Bits 15:8  | TDB1[7:0] | 8-bit transmit data (next byte after TDB0[7:0] on I3C bus). |
| Bits 7:0   | TDB0[7:0] | 8-bit transmit data (earliest byte on I3C bus) |
52.16.8  I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020
Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:
- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:
- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIIRD[2:0] in the I3C_RMR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24  IBIDB3[7:0]: 8-bit IBI payload data (latest byte on I3C bus).
Bits 23:16  IBIDB2[7:0]: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).
Bits 15:8  IBIDB1[7:0]: 8-bit IBI payload data (next byte on I3C bus after IBIDBO[7:0]).
Bits 7:0   IBIDBO[7:0]: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).
52.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024
Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFRG register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFRG register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)
- This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.
- This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.
  - 0: no TX-FIFO preload
  - 1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)
- This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.
- This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.
52.16.10 I3C status register (I3C_SR)

Address offset: 0x030
Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 31:24 M[7:0]: Message identifier/counter of a given frame (when the I3C acts as controller)
When I3C acts as controller, this bitfield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.
First message of a frame is identified with M[7:0] = 0.
This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 DIR: Message direction
Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus
0: write
1: read

Note: ENTDAA CCC is considered as a write command.
Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target
1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)
- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message
- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

**52.16.11 I3C status error register (I3C_SER)**

Address offset: 0x034
Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error
1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error
1: controller detected that a data byte is not acknowledged by a target, either during:
   i) a legacy I2C write transfer
   ii) the second trial when sending dynamic address during ENTDAA procedure
Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)
   0: no detected error
   1: controller detected that the static/dynamic address was not acknowledged by a target,
      either during:
      i) a legacy I2C read/write transfer
      ii) a direct CCC write transfer
      iii) the second trial of a direct CCC read transfer
      iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)
   0: no detected error
   1: controller detected either:
      i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted
      ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun
   0: no detected error
   1: whatever controller or target, hardware detected either:
      i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted
      ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)
   0: no detected error
   1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read
      (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error
   0: no detected error
   1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]
52.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

| Bits 31:24 | Reserved, must be kept at reset value. |
| Bits 23:17 | **RADD[6:0]**: Received target address (when the I3C is configured as controller) |
| When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request. |

| Bit 16 | Reserved, must be kept at reset value. |

---

**Improved inter-integrated circuit (I3C)**

**RM0477**

---

**I3C received message register (I3C_RMR)**

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

| Bits 31:24 | Reserved, must be kept at reset value. |
| Bits 23:17 | **RADD[6:0]**: Received target address (when the I3C is configured as controller) |
| When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request. |

| Bit 16 | Reserved, must be kept at reset value. |
Improved inter-integrated circuit (I3C)

Bits 15:8 RCODE[7:0]: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

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

Bits 2:0 IBIRDCNT[2:0]: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

52.16.13 I3C event register (I3C_EVR)

Address offset: 0x050
Reset value: 0x0000 0003
This is a read register, used for reporting event flags.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRPF</td>
<td>DEFF</td>
<td>INT</td>
<td>UPDF</td>
<td>AS</td>
<td>RSTF</td>
<td>MRL</td>
<td>MVL</td>
<td>DA</td>
<td>STAF</td>
<td>GETF</td>
<td>WKPFE</td>
<td>HJF</td>
<td>CR</td>
<td>UPDF</td>
<td>CRF</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IBIF</th>
<th>Res.</th>
<th>Res.</th>
<th>Res.</th>
<th>ERRF</th>
<th>RXGT</th>
<th>ENDF</th>
<th>FCF</th>
<th>Res.</th>
<th>RX</th>
<th>LASTF</th>
<th>RX</th>
<th>FNEF</th>
<th>TX</th>
<th>LASTF</th>
<th>RX</th>
<th>FNEF</th>
<th>TX</th>
<th>FNFF</th>
<th>SFNFE</th>
<th>CFNFE</th>
<th>TXEF</th>
<th>CFEF</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bit 31 GRPF: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role. The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 DEFF: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role. The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 INTUPDF: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register. The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 ASUPDF: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register. The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.
Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)
When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop). Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.
- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.
When in Stop mode, the corresponding interrupt can be used to wake up the device.
The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.
The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.
The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.
The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.
The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.
The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.
Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.
The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.
The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.
The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.
The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.
The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).
The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)
When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.
The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)
This flag is asserted by hardware to indicate that an error occurred. Then, software must read I3C_SER to get the error type.
The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.
Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.
Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

*Note*: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty. This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target. This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

*Note*: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFRG register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

### 52.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event(flag).

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>GRPIE: DEFGRPA CCC interrupt enable (when the I3C acts as target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>DEFIE: DEFTGTS CCC interrupt enable (when the I3C acts as target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>INTUPDIE: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>interrupt disabled</td>
</tr>
<tr>
<td>1</td>
<td>interrupt enabled</td>
</tr>
</tbody>
</table>
Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 20 **Reserved, must be kept at reset value.**

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
   0: interrupt disabled
   1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
   0: interrupt disabled
   1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
   0: interrupt disabled
   1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
   0: interrupt disabled
   1: interrupt enabled

Bits 14:12 **Reserved, must be kept at reset value.**
Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
   0: interrupt disabled
   1: interrupt enabled

Bit 10 **RXTGTDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
   0: interrupt disabled
   1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
   0: interrupt disabled
   1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
   0: interrupt disabled
   1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
   0: interrupt disabled
   1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
   0: interrupt disabled
   1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
   0: interrupt disabled
   1: interrupt enabled

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

### 52.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_CEVR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGPF</td>
<td>CDEFF</td>
<td>CINTLCPF</td>
<td>CASLPDF</td>
<td>CRSTF</td>
<td>CMRLUPDF</td>
<td>CMWLPDF</td>
<td>CDAUPDF</td>
<td>CSTAF</td>
<td>CGETF</td>
<td>CWKPF</td>
<td>CRSTF</td>
<td>CMRLUPDF</td>
<td>CMWLPDF</td>
<td>CDAUPDF</td>
<td>CCRUPDF</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w | w |

Bit 31 **CGRPF**: Clear DEGFRPA CCC flag (when the I3C acts as target)
   0: no effect
   1: clear GRPF
Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)

0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)

0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)

0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)

0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)

0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)

0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)

0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)

0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)

0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)

0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)

0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)

0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)

0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)

0: no effect
1: clear IBIENDF
Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IIBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

### 52.16.16 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RSTACT[1:0]</td>
<td>Target reset action</td>
<td>rw</td>
<td>RSTACT[1:0]</td>
</tr>
<tr>
<td>30</td>
<td>AS[1:0]</td>
<td>Address space</td>
<td>r</td>
<td>AS[1:0]</td>
</tr>
<tr>
<td>29</td>
<td>HJEN</td>
<td>Hardware join enable</td>
<td>rw</td>
<td>HJEN</td>
</tr>
<tr>
<td>28</td>
<td>CREN</td>
<td>Clear request enable</td>
<td>rw</td>
<td>CREN</td>
</tr>
<tr>
<td>27</td>
<td>IBIEN</td>
<td>Interrupt enable</td>
<td>rw</td>
<td>IBIEN</td>
</tr>
<tr>
<td>26</td>
<td>RSTVAL</td>
<td>Reset action valid</td>
<td>r</td>
<td>RSTVAL</td>
</tr>
<tr>
<td>25</td>
<td>RSTACT[1:0]</td>
<td>Target reset action</td>
<td>rw</td>
<td>RSTACT[1:0]</td>
</tr>
<tr>
<td>24</td>
<td>DA[6:0]</td>
<td>Dynamic address</td>
<td>rw</td>
<td>DA[6:0]</td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)
This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid. This bit is cleared by hardware when the target receives a frame start.

When the device is not in Stop mode:
- If RSTVAL = 1: when RSTF in the I3C_EVIR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.
Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus. This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

- **00**: no reset action
- **01**: first level of reset: the application software must either:
  - a) partially reset the peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).
  - b) fully reset the peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

- **10**: second level of reset: the application software must issue a warm reset, also known as a system reset. This (see [Section 7: Reset and clock control (RCC)](#)) has the same impact as a pin reset (NRST = 0):
  - the software writes and sets the SYSRESETREQ control bit of the AITR register, when the device is controlled by a Cortex®-M.
  - the software writes and sets SYSRST = 1 in the RCC_ISRSTCSETR register, when the device is controlled by a Cortex®-A.

- **11**: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

- **00**: activity state 0
- **01**: activity state 1
- **10**: activity state 2
- **11**: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

- **0**: hot-join request disabled
- **1**: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

- **0**: controller-role request disabled
- **1**: controller-role request enabled
Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGRx register.

0: IBI request disabled
1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

52.16.17 **I3C device x characteristics register (I3C_DEVRx)**

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVRx register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bit 31 DIS**: DA[6:0] write disabled (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS = 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS = 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed
1: write to DA[7:0] and to IBIDEN is disabled/locked
Bits 30:20  Reserved, must be kept at reset value.

Bit 19  **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)
When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).
If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.
0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted
1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18  **IBIDEN**: IBI data enable (when the I3C acts as controller)
When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.
Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.
0: no data byte follows the acknowledged IBI from target x
1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17  **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)
When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:
0: a controller-role request from target x must be NACK-ed
   After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)
1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed
   - The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.
   - After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.
   - Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.
52.16.18 **I3C maximum read length register (I3C_MAXRLR)**

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFG register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

IBIP[2:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

31:19 Reserved, must be kept at reset value.
52.16.19 **I3C maximum write length register (I3C_MAXWLR)**

Address offset: 0x094
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>IBIP[2:0]: IBI payload data maximum size, in bytes (when I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged. This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).</td>
</tr>
<tr>
<td></td>
<td>Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.</td>
</tr>
<tr>
<td></td>
<td>000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)</td>
</tr>
<tr>
<td></td>
<td>001: 1 byte (mandatory data byte MDB[7:0])</td>
</tr>
<tr>
<td></td>
<td>010: 2 bytes (including first MDB[7:0])</td>
</tr>
<tr>
<td></td>
<td>011: 3 bytes (including first MDB[7:0])</td>
</tr>
<tr>
<td></td>
<td>100: 4 bytes (including first MDB[7:0])</td>
</tr>
<tr>
<td></td>
<td>others: same as 100</td>
</tr>
<tr>
<td>15-0</td>
<td>MRL[15:0]: Maximum data read length (when I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).</td>
</tr>
<tr>
<td></td>
<td>Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.</td>
</tr>
<tr>
<td></td>
<td>This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15-0</td>
<td>MWL[15:0]: Maximum data write length (when I3C acts as target)</td>
</tr>
<tr>
<td></td>
<td>This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.</td>
</tr>
<tr>
<td></td>
<td>Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.</td>
</tr>
<tr>
<td></td>
<td>This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.</td>
</tr>
</tbody>
</table>
52.16.20 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0
Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 SCLH_I2C[7:0]: SCL high duration, used for legacy I2C messages, in number of kernel clocks cycles:

\[ t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK} \]

Note: SCLH_I2C is used to generate \( t_{DIG_H} \) (I2C) timing when communicating with I2C devices.

Note: With I2C fm+ device \( t_{DIG_H_{min}} = 260 \) ns, with I2C fm device \( t_{DIG_H_{min}} = 600 \) ns.

Bits 23:16 SCLL_OD[7:0]: SCL low duration in open-drain phases, used for legacy I2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

\[ t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK} \]

Note: SCLL_OD is used to generate both \( t_{DIG_L} \) (I2C) and \( t_{DIG_OD_L} \) (I3C) timings.

Note: With I2C fm+ device \( t_{DIG_L_{min}} = 500 \) ns, with I2C fm device \( t_{DIG_L_{min}} = 1320 \) ns.

Note: I3C messages: \( t_{DIG_OD_L_{min}} = 200 \) ns.

Note: If a single I3C frame is gathering I2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I2C timings.

Bits 15:8 SCLH_I3C[7:0]: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

\[ t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK} \]

Note: SCLH_I3C is used to generate both \( t_{DIG_H} \) (I3C) and \( t_{DIG_H_{MIXED}} \) timings.

Note: For mixed bus (with at least one I2C target): \( t_{DIG_H_{MIXED_{min}}} = 32 \) ns and \( t_{DIG_H_{MIXED_{max}}} = 45 \) ns (due to I2C 50 ns spike filter).

Note: For pure I3C bus (with no I2C targets): \( t_{DIG_{H_{min}}} = 32 \) ns.

Bits 7:0 SCLL_PP[7:0]: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

\[ t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK} \]

Note: SCLL_PP is used to generate \( t_{DIG_L} \) (I3C in PP) timing.

Note: \( t_{DIG_L_{min}} = 32 \) ns (max 40/60 duty cycle at 12.5 MHz).
52.16.21 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4
Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **SDA_HD**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull $t_{HD_{PP}}$):

$SDA\text{ hold time} = (SDA\_HD + 0.5) \times t_{I3CCLK}$

*Note:* when controller: $t_{HD_{PPmin}} = min\{t_{CR}, t_{CF}\} + 3$ ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) $t_{CAS}$ and (I2C) $t_{BUF}$. These timings are defined by: $t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) \times 2 - (0.5 + SDA\_HD)] \times t_{I3CCLK}$

*Note:* For pure I3C bus: $t_{CAS}\text{min} = 38.4$ ns, and $t_{CAS}\text{max} = 1$ µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

*Note:* For mixed bus with $I^2C$ fm+ device $t_{BUF\text{min}} = 0.5$ µs, for mixed bus with $I^2C$ fm device

$t_{BUF\text{min}} = 1.3$ µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to MIPI timing $t_{CASr}$. This timing is defined by: $t_{CASr} = [(FREE[6:0] + 1) \times 2 - (0.5 + SDA\_HD)] \times t_{I3CCLK}$

*Note:* $t_{CASr, min} = 19.2$ ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by:

$t_{CBP} = (FREE[6:0] + 1) \times t_{I3CCLK}$

*Note:* $t_{CBP\text{min}} = 19.2$ ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by:

$t_{CBSr} = (FREE[6:0] + 1) \times t_{I3CCLK}$

*Note:* $t_{CBSr, min} = 19.2$ ns.

Bits 15:10 Reserved, must be kept at reset value.
Bits 9:8 **ASNCR[1:0]:** Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0]. This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]:** Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This field is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

- **When the I3C acts as target:**
  1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t\textsubscript{AVAL}). Refer to MIPI timing t\textsubscript{AVAL} = 1 µs. This timing is defined by: t\textsubscript{AVAL} = (AVAL[7:0] + 2) x t\textsubscript{I3CCLK}
  2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t\textsubscript{IDLE} = 200 µs. This timing is defined by: t\textsubscript{IDLE} = (AVAL[7:0] + 2) x 200 x t\textsubscript{I3CCLK}

- **When the I3C acts as controller,** it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:
  1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing t\textsubscript{STALLDAA} = 15 ms. This timing is defined by:
     t\textsubscript{STALLDAA}\textsuperscript{max} = (AVAL[7:0] + 1) x 15000 x t\textsubscript{I3CCLK}
  2. on ACK/NACK phase of I3C/I2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t\textsubscript{STALL} = 100 µs. This timing is defined by: t\textsubscript{STALL}\textsuperscript{max} = (AVAL[7:0] + 1) x 100 x t\textsubscript{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for t\textsubscript{NEWCRLock} before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t\textsubscript{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same t\textsubscript{NEWCRLock} time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ASNCR[1:0] in the I3C\_TIMINGR1 register:
   - ASNCR[1:0] = 00: t\textsubscript{NEWCRLock} = (AVAL[7:0] + 1) x t\textsubscript{I3CCLK}
   - ASNCR[1:0] = 01: t\textsubscript{NEWCRLock} = (AVAL[7:0] + 1) x 100 x t\textsubscript{I3CCLK}
   - ASNCR[1:0] = 10: t\textsubscript{NEWCRLock} = (AVAL[7:0] + 1) x 2000 x t\textsubscript{I3CCLK}
   - ASNCR[1:0] = 11: t\textsubscript{NEWCRLock} = (AVAL[7:0] + 1) x 50000 x t\textsubscript{I3CCLK}
52.16.22 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8
Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I2C target(s).

This register can be written only when the I3C acts as controller.

### I3C TIMINGR2 Register

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bits 15:8</th>
<th>Bits 7:4</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved, must be kept at reset value.</td>
<td><strong>STALL[7:0]</strong></td>
<td>Reserved, must be kept at reset value.</td>
<td><strong>STALLT</strong></td>
</tr>
</tbody>
</table>

**STALL[7:0]: Controller clock stall time, in number of kernel clock cycles**

\[ t_{SCLL\_STALL} = STALL \times t_{I3C\_CLK} \]

**Bit 3 ** **STALLA**: Controller clock stall enable on ACK phase

The SCL is stalled (during \( t_{SCLL\_STALL} \)) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall
1: stall enabled

**Bit 2 ** **STALLC**: Controller clock stall enable on PAR phase of CCC

The SCL is stalled during \( STALL \times t_{SCLL\_PP} \) in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall
1: stall enabled

**Note:** \( t_{SCLL\_PP} = (I3C\_TIMINGR0.SCLL\_PP[7:0] + 1) \times t_{I3C\_CLK} \)

**Bit 1 ** **STALLD**: Controller clock stall enable on PAR phase of Data

The SCL is stalled during \( STALL \times t_{SCLL\_PP} \) in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall
1: stall enabled

**Note:** \( t_{SCLL\_PP} = (I3C\_TIMINGR0.SCLL\_PP[7:0] + 1) \times t_{I3C\_CLK} \)

**Bit 0 ** **STALLT**: Controller clock stall enable on T-bit phase of data (and on the ACK/NACK phase of data byte of a legacy I2C read)

The SCL is stalled during \( STALL \times t_{SCLL\_PP} \) in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall
1: stall enabled

**Note:** \( t_{SCLL\_PP} = (I3C\_TIMINGR0.SCLL\_PP[7:0] + 1) \times t_{I3C\_CLK} \)
52.16.23 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable
0: I3C target (no controller capable)
1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.
### 52.16.24 I3C device characteristics register (I3C_DCR)

- **Address offset:** 0x0C4
- **Reset value:** 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFR register.

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td>DCR[7:0]: device characteristics ID</td>
</tr>
<tr>
<td>0x00</td>
<td>generic device (for v1.0 devices)</td>
</tr>
<tr>
<td>others</td>
<td>ID to describe the type of the I3C sensor/device</td>
</tr>
</tbody>
</table>

*Note: The latest MIPI DCR ID assignments are available on [https://www.mipi.org](https://www.mipi.org).*
52.16.25 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGMR register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14  **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

- 0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification
- 1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.
52.16.26 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:
- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRCAP2[7:0] on the I3C bus is then as follows:
- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFG register.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Cap Group Capability (when acting as controller)</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>CAPGRP: group management support (when acting as controller)</td>
</tr>
<tr>
<td>28</td>
<td>This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.</td>
</tr>
<tr>
<td>27</td>
<td>This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.</td>
</tr>
<tr>
<td>26</td>
<td>0: this I3C does not support group address capabilities</td>
</tr>
<tr>
<td>25</td>
<td>1: this I3C supports group address capabilities (when becoming controller)</td>
</tr>
<tr>
<td>24</td>
<td>Cap Group Delayed Controller-Role Handoff (when acting as controller)</td>
</tr>
<tr>
<td>23</td>
<td>Bit 3 CAPDHOFF: delayed controller-role hand-off</td>
</tr>
<tr>
<td>22</td>
<td>This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.</td>
</tr>
<tr>
<td>21</td>
<td>This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.</td>
</tr>
<tr>
<td>20</td>
<td>0: this I3C does not needs additional time to process a controller-role hand-off</td>
</tr>
<tr>
<td>19</td>
<td>1: this I3C needs additional time to process a controller-role hand-off</td>
</tr>
</tbody>
</table>

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

Bit 9 CAPGRP: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities
1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 CAPDHOFF: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off
1: this I3C needs additional time to process a controller-role hand-off
52.16.27 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0
Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:
- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:
- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time \( t_{SCO} \) \( \leq 12 \) ns) in the I3C_GETMXDSR register, else 111 (\( t_{SCO} > 12 \) ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:
- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:
- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFG register.
Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **TSCO**: clock-to-data turnaround time ($t_{SCO}$)
This bit is written by software during bus initialization (EN = 0 in the I3C_CFRGR register) and is used to specify the clock-to-data turnaround time $t_{SCO}$ vs. the value of 12 ns. This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.
- 0: $t_{SCO} \leq 12$ ns
- 1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16  **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdT (maximum read turnaround byte)
This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdT, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:8  **FMT[1:0]**: GETMXDS CCC format
This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdT) and format 2 (with MaxRdT).
This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdT) byte in response to the GETCAPS CCC.
- 00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)
- 01: format 2: (5 bytes with MaxWr with no defining byte, MaxRd, MaxRdT)
  - 3-byte MaxRdT is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
  - Max read turnaround time is less than 256 µs.
- 10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdT)
  - 3-byte MaxRdT is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
  - Max read turnaround time is between 256 and 65535 µs
- 11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdT)
  - 3-byte MaxRdT is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
  - Max read turnaround time is between 65535 µs and 16 s.

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

Bits 1:0  **HOFFAS[1:0]**: Controller hand-off activity state
This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.
This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.
- 00: activity state 0 is the initial activity state of this I3C before and when becoming controller
- 01: activity state 1 is the initial activity state of this I3C when becoming controller
- 10: activity state 2 is the initial activity state of this I3C when becoming controller
- 11: activity state 3 is the initial activity state of this I3C when becoming controller
52.16.28  **I3C extended provisioned ID register (I3C_EPIDR)**

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>I3C_CR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Add[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Addr[6:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CCC[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>DCNT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:17  **MIPIMID[14:0]**  15-bit MIPI manufacturer ID

This read field is the 15-bit STMicroelectronics MIPI ID (0x0104). This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16  **IDTSEL**: provisioned ID type selector

This field is set as 0 (vendor fixed value). This field represents bit[32] of the 48-bit provisioned ID.

*Note: Bits[31:16] of the provisioned ID can be 0.*

Bits 15:12  **MIPIID[3:0]**  4-bit MIPI Instance ID

This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus. This field represents bits[15:12] of the 48-bit provisioned ID.

*Note: Bits[11:0] of the provisioned ID can be 0.*

Bits 11:0  Reserved, must be kept at reset value.

52.16.29  **I3C register map**

**Table 552. I3C register map and reset values**

| Offset | Register name | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|---------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | I3C_CR        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

Reset value: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
### Table 552. I3C register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>I3C_CFGR</td>
<td>0x004</td>
<td>I3C_CFGR</td>
<td>0x004</td>
<td>I3C_CFGR</td>
<td>0x004</td>
<td>I3C_CFGR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x010</td>
<td>I3C_RDR</td>
<td>0x018</td>
<td>I3C_TDR</td>
<td>0x01C</td>
<td>I3C_TDR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>I3C_RDONLY</td>
<td>0x014</td>
<td>I3C_RDONLY</td>
<td>0x014</td>
<td>I3C_RDONLY</td>
<td>0x014</td>
<td>I3C_RDONLY</td>
</tr>
<tr>
<td></td>
<td>RDB[7:0]</td>
<td></td>
<td>RDB[7:0]</td>
<td></td>
<td>RDB[7:0]</td>
<td></td>
<td>RDB[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x018</td>
<td>I3C_RDWR</td>
<td>0x018</td>
<td>I3C_RDWR</td>
<td>0x018</td>
<td>I3C_RDWR</td>
</tr>
<tr>
<td></td>
<td>RDB3[7:0]</td>
<td></td>
<td>RDB2[7:0]</td>
<td></td>
<td>RDB1[7:0]</td>
<td></td>
<td>RDB0[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x01C</td>
<td>I3C_TWRD</td>
<td>0x01C</td>
<td>I3C_TWRD</td>
<td>0x01C</td>
<td>I3C_TWRD</td>
</tr>
<tr>
<td></td>
<td>TDB[7:0]</td>
<td></td>
<td>TDB[7:0]</td>
<td></td>
<td>TDB[7:0]</td>
<td></td>
<td>TDB[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x020</td>
<td>I3C_IBIDR</td>
<td>0x020</td>
<td>I3C_IBIDR</td>
<td>0x020</td>
<td>I3C_IBIDR</td>
</tr>
<tr>
<td></td>
<td>IBID3[7:0]</td>
<td></td>
<td>IBID2[7:0]</td>
<td></td>
<td>IBID1[7:0]</td>
<td></td>
<td>IBID0[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x024</td>
<td>I3C_TGTTDR</td>
<td>0x024</td>
<td>I3C_TGTTDR</td>
<td>0x024</td>
<td>I3C_TGTTDR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x030</td>
<td>I3C_SR</td>
<td>0x030</td>
<td>I3C_SR</td>
<td>0x030</td>
<td>I3C_SR</td>
</tr>
<tr>
<td></td>
<td>MID[7:0]</td>
<td></td>
<td>MID[7:0]</td>
<td></td>
<td>MID[7:0]</td>
<td></td>
<td>MID[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x034</td>
<td>I3C_SER</td>
<td>0x034</td>
<td>I3C_SER</td>
<td>0x034</td>
<td>I3C_SER</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x040</td>
<td>I3C_RMR</td>
<td>0x040</td>
<td>I3C_RMR</td>
<td>0x040</td>
<td>I3C_RMR</td>
</tr>
<tr>
<td></td>
<td>RCODE[7:0]</td>
<td></td>
<td>RCODE[7:0]</td>
<td></td>
<td>RCODE[7:0]</td>
<td></td>
<td>RCODE[7:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x050</td>
<td>I3C_EVR</td>
<td>0x050</td>
<td>I3C_EVR</td>
<td>0x050</td>
<td>I3C_EVR</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0x054</td>
<td>I3C_IER</td>
<td>0x054</td>
<td>I3C_IER</td>
<td>0x054</td>
<td>I3C_IER</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
</tr>
<tr>
<td>----------</td>
<td>---------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>0x058</td>
<td>I3C_CEVR</td>
<td>G0</td>
<td>G0</td>
<td>G0</td>
<td>G0</td>
<td>G0</td>
<td>G0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RSTVAL</td>
<td>RSTACT</td>
<td>[1:0]</td>
<td>AS</td>
<td>[10]</td>
<td>HJEN</td>
</tr>
<tr>
<td>0x05C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060</td>
<td>I3C_DEVR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x060 + 4 * x, (x = 1 to 4)</td>
<td>I3C_DEVRx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SUSP</td>
<td>BIDEN</td>
<td>CRACK</td>
<td>IBACK</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x090</td>
<td>I3C_MAXRLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x094</td>
<td>I3C_MAXWLR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0A0</td>
<td>I3C_TIMINGR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0A4</td>
<td>I3C_TIMINGR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0A8</td>
<td>I3C_TIMINGR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C0</td>
<td>I3C_BCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C4</td>
<td>I3C_DCR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C8</td>
<td>I3C_GETCAPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0CC</td>
<td>I3C_CRCAPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 552. I3C register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0D0  | I3C_GETMXDSR  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |   | Reset value | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
| 0x0D4  | I3C_EPIDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |   | MIPIMID[14:0] | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
|        |   | MIPIID[3:0]  | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
|        |   | IDTSEL       | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |

Refer to Section 2.3 for the register boundary addresses.
53 Universal synchronous/asynchronous receiver transmitter (USART/UART)

This section describes the universal synchronous asynchronous receiver transmitter (USART/UART).

53.1 Introduction

The USART offers a flexible means to perform Full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

53.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data
  - Each FIFO can be enabled/disabled by software and come with a status flag.
  - A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
• Communication control/error detection flags
• Parity control:
  – Transmits parity bit
  – Checks parity of received data byte
• Interrupt sources with flags
• Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
• Wake-up from Stop mode

53.3 USART extended features
• LIN master synchronous break send capability and LIN slave break detection capability
  – 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
• IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
• Smartcard mode
  – Supports the T=0 and T=1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  – 0.5 and 1.5 stop bits for smartcard operation
• Support for Modbus communication
  – Timeout feature
  – CR/LF character recognition

53.4 USART implementation
The tables below describe USART implementation. It also includes LPUART for comparison.

<table>
<thead>
<tr>
<th>Instance</th>
<th>STM32H7Rx/7Sx</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART1</td>
<td>Full</td>
</tr>
<tr>
<td>USART2</td>
<td>Full</td>
</tr>
<tr>
<td>USART3</td>
<td>Full</td>
</tr>
<tr>
<td>UART4</td>
<td>Basic</td>
</tr>
<tr>
<td>UART5</td>
<td>Basic</td>
</tr>
<tr>
<td>UART7</td>
<td>Basic</td>
</tr>
<tr>
<td>UART8</td>
<td>Basic</td>
</tr>
<tr>
<td>LPUART1</td>
<td>Low-power</td>
</tr>
</tbody>
</table>
Table 554. USART/LPUART features

<table>
<thead>
<tr>
<th>Modes/features&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Full feature set</th>
<th>Basic feature set</th>
<th>Low-power feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware flow control for modem</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Synchronous mode (master/slave)</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Smartcard mode</td>
<td>X</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Single-wire half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td></td>
<td>7, 8 and 9 bits</td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size (bytes)</td>
<td></td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

1. "X" = supported, '-' = not supported.
2. Wake-up supported from Stop mode.
53.5 USART functional description

53.5.1 USART block diagram

Figure 733. USART block diagram

53.5.2 USART pins and internal signals

Description USART input/output pins

- USART bidirectional communications
  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):
    - RX (Receive Data Input)
      RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.
    - TX (Transmit Data Output)
      When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.
• RS232 hardware flow control mode
  The following pins are required in RS232 hardware flow control mode:
  – CTS (Clear To Send)
    When driven high, this signal blocks the data transmission at the end of the current transfer.
  – RTS (Request To Send)
    When it is low, this signal indicates that the USART is ready to receive data.

• RS485 hardware control mode
  The DE (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

• Synchronous master/slave mode and smartcard mode
  The following pins are required in synchronous master/slave mode and smartcard mode:
  – CK
    This pin acts as Clock output in synchronous master and smartcard modes. It acts as Clock input in synchronous slave mode.
    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.
    In smartcard mode, CK output provides the clock to the smartcard.
  – NSS
    This pin acts as Slave Select input in synchronous slave mode.

Refer to Table 555 and Table 556 for the list of USART input/output pins and internal signals.

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART_RX/UART_RX</td>
<td>Input</td>
<td>Serial data receive input</td>
</tr>
<tr>
<td>USART_TX/UART_TX</td>
<td>Output</td>
<td>Transmit data output</td>
</tr>
<tr>
<td>USART_CTS/UART_CTS</td>
<td>Input</td>
<td>Clear to send</td>
</tr>
<tr>
<td>USART_RTS/UART_RTS</td>
<td>Output</td>
<td>Request to send</td>
</tr>
<tr>
<td>USART_DE(1)/UART_DE(2)</td>
<td>Output</td>
<td>Driver enable</td>
</tr>
<tr>
<td>USART_CK</td>
<td>Output</td>
<td>Clock output in synchronous master and smartcard modes.</td>
</tr>
<tr>
<td>USART_NSS(3)</td>
<td>Input</td>
<td>Slave select input in synchronous slave mode.</td>
</tr>
</tbody>
</table>

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.
Description of USART input/output signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usart_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
<tr>
<td>usart_ker_ck</td>
<td>Input</td>
<td>USART kernel clock</td>
</tr>
<tr>
<td>usart_wkup</td>
<td>Output</td>
<td>USART provides a wake-up interrupt</td>
</tr>
<tr>
<td>usart_it</td>
<td>Output</td>
<td>USART global interrupt</td>
</tr>
<tr>
<td>usart_tx_dma</td>
<td>Input/output</td>
<td>USART transmit DMA request</td>
</tr>
<tr>
<td>usart_rx_dma</td>
<td>Input/output</td>
<td>USART receive DMA request</td>
</tr>
</tbody>
</table>

### 53.5.3 USART clocks

The simplified block diagram given in Section 53.5.1: USART block diagram shows two fully-independent clock domains:

- The **usart_pclk** clock domain
  
  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.

- The **usart_ker_ck** kernel clock domain.
  
  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker ck** clock is the same as the **usart_pclk** clock.

  There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

  When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

### 53.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see Figure 734):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

**Note:** In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.
An *Idle character* is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A *Break character* is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.
Figure 734. Word length programming

9-bit word length (M = 01), 1 Stop bit

8-bit word length (M = 00), 1 Stop bit

7-bit word length (M = 10), 1 Stop bit

** LBCL bit controls last data clock pulse
53.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields. In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG data have been received: 1 data in USART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size -1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

53.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.
**Universal synchronous/asynchronous receiver transmitter (USART/UART) **

**Note:** The TE bit must be set before writing the data to be transmitted to the USART_TDR. The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost. An idle frame is sent when the TE bit is enabled.

**Configurable stop bits**

The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13, 12.

- **1 stop bit:** This is the default value of number of stop bits.
- **2 stop bits:** This is supported by normal USART, single-wire and modem modes.
- **1.5 stop bits:** To be used in smcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see Section 53.5.1: USART block diagram). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

**Figure 735. Configurable stop bits**

**Character transmission procedure**

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in Section 53.5.20: Continuous communication using USART and DMA.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   - When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.

8. When the last data is written to the USART_TDR register, wait until TC = 1.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

**Single byte communication**

- When FIFO mode is disabled
  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  - the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data.

This flag generates an interrupt if the TXEIE bit is set.

When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFN (TXFIFO not full) flag is set by hardware to indicate that:
  - the TXFIFO is not full;
  - the USART_TDR register is empty;
  - the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNIE bit is set.

Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.
After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see Figure 736: TC/TXE behavior when transmitting).

**Figure 736. TC/TXE behavior when transmitting**

---

**Note:** When FIFO management is enabled, the TXFNF flag is used for data transmission.

### Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see Figure 734).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

### Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

### 53.5.7 USART receiver

The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

#### Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.
In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0 X 0X 0.

**Figure 737. Start bit detection when oversampling by 16 or 8**

<table>
<thead>
<tr>
<th>RX state</th>
<th>Idle</th>
<th>Start bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX line</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal sample clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Real sample clock</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditions to validate the start bit</td>
<td>1 1 0 X</td>
<td>X X X X</td>
</tr>
<tr>
<td>7/16</td>
<td>6/16</td>
<td>One-bit time</td>
</tr>
<tr>
<td>0 0</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>0 0</td>
<td>X X X X</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE=1, or RXFNE flag set and interrupt generated if RXFNEIE=1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).
Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in Section 53.5.20: Continuous communication using USART and DMA.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be
generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the USART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

**Overrun error**

- **FIFO mode disabled**
  
  An overrun error occurs if a character is received and RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception. An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  
  - the ORE bit is set;
  
  - the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  
  - the shift register is overwritten. After that, any data received during overrun is lost.
  
  - an interrupt is generated if either the RXNEIE or the EIE bit is set.

- **FIFO mode enabled**
  
  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full. Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty. An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  
  - The ORE bit is set.
  
  - The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  
  - The shift register is overwritten. After that point, any data received during overrun is lost.
  
  - An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the OREC bit in the USART_ICR register.

*Note:* The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,

- if RXNE=0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.
Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see Section: Reset and Clock Control (RCC)). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is `usart_ker_ck`.

When the dual clock domain and the wake-up from low-power mode features are supported, the `usart_ker_ck` clock source can be configurable in the RCC (see Section: Reset and Clock Control (RCC)). Otherwise the `usart_ker_ck` clock is the same as `usart_pclk`.

The `usart_ker_ck` clock can be divided by a programmable factor, defined in the `USART_PRESC` register.

Some `usart_ker_ck` sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the `USART_RDR` register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the `USART_CR1` register either to 16 or 8 times the baud rate clock (see Figure 739 and Figure 740).

Depending on the application:
- select oversampling by 8 (OVER8=1) to achieve higher speed (up to `usart_ker_ck_pres/8`). In this case the maximum receiver tolerance to clock deviation is reduced (refer to Section 53.5.9: Tolerance of the USART receiver to clock deviation on page 2427)
- select oversampling by 16 (OVER8=0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum
Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.

- A single sample in the center of the received bit

Depending on the application:
- select the three sample majority vote method (ONEBIT=0) when operating in a noisy environment and reject the data when a noise is detected (refer to Table 557) because this indicates that a glitch occurred during the sampling.
- select the single sample method (ONEBIT=1) when the line is noise-free to increase the receiver tolerance to clock deviations (see Section 53.5.9: Tolerance of the USART receiver to clock deviation on page 2427). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI mode.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.
Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:
- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled);
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.
Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

• **0.5 stop bit (reception in smartcard mode):** no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.

• **1 stop bit:** sampling for 1 stop bit is done on the 8th, 9th and 10th samples.

• **1.5 stop bits (smartcard mode):**
  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE =1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to Section 53.5.17: USART receiver timeout on page 2440 for more details).

• **2 stop bits:**
  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit. The framing error flag is set if a framing error is detected during the first stop bit. The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

53.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

**Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)**

In case of oversampling by 16, the baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{\text{uart_ker_ck_pres}}{\text{USARTDIV}}
\]

In case of oversampling by 8, the baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{2 \times \text{uart_ker_ck_pres}}{\text{USARTDIV}}
\]

**Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)**

The baud rate is given by the following formula:

\[
\text{Tx/Rx baud} = \frac{\text{uart_ker_ck_pres}}{\text{USARTDIV}}
\]
USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1
To obtain 9600 bauds with uart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  \[ USARTDIV = \frac{8\ 000\ 000}{9600} \]
  \[ BRR = USARTDIV = 0d833 = 0x0341 \]

- In case of oversampling by 8:
  \[ USARTDIV = \frac{2 \times 8\ 000\ 000}{9600} \]
  \[ USARTDIV = 1666.66 (0d1667 = 0x683) \]
  \[ BRR[3:0] = 0x3 >> 1 = 0x1 \]
  \[ BRR = 0x681 \]

Example 2
To obtain 921.6 kbauds with uart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  \[ USARTDIV = \frac{48\ 000\ 000}{921\ 600} \]
  \[ BRR = USARTDIV = 0x52 = 0x34 \]

- In case of oversampling by 8:
  \[ USARTDIV = 2 \times 48\ 000\ 000/921\ 600 \]
  \[ USARTDIV = 104 (0d104 = 0x68) \]
  \[ BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4 \]
  \[ BRR = 0x64 \]
53.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- **DTRA**: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- **DQUANT**: error due to the baud rate quantization of the receiver
- **DREC**: deviation of the receiver local oscillator
- **DTCL**: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[ \text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{USART receiver tolerance} \]

where

\[ \text{DWU} \text{ is the error due to sampling point deviation when the wake-up from low-power mode is used.} \]

- when \( \text{M}[1:0] = 01 \):
  \[ \text{DWU} = \frac{t_{\text{WUUSART}}}{11 \times \text{Tbit}} \]
- when \( \text{M}[1:0] = 00 \):
  \[ \text{DWU} = \frac{t_{\text{WUUSART}}}{10 \times \text{Tbit}} \]
- when \( \text{M}[1:0] = 10 \):
  \[ \text{DWU} = \frac{t_{\text{WUUSART}}}{9 \times \text{Tbit}} \]

\( t_{\text{WUUSART}} \) is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 558, Table 559, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.
53.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from $\frac{\text{usart\_ker\_ck\_pres}}{65535}$ and $\frac{\text{usart\_ker\_ck\_pres}}{16}$.
- When oversampling by 8, the baud rate ranges from $\frac{\text{usart\_ker\_ck\_pres}}{32763}$ and $\frac{\text{usart\_ker\_ck\_pres}}{8}$.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

Note: The data specified in Table 558 and Table 559 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when $M$ bits = 00 (11-bit times when $M$ = 01 or 9-bit times when $M$ = 10).

### Table 558. Tolerance of the USART receiver when BRR [3:0] = 0000

<table>
<thead>
<tr>
<th>M bits</th>
<th>OVER8 bit = 0</th>
<th>OVER8 bit = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ONEBIT=0</td>
<td>ONEBIT=1</td>
</tr>
<tr>
<td>00</td>
<td>3.75%</td>
<td>4.375%</td>
</tr>
<tr>
<td>01</td>
<td>3.41%</td>
<td>3.97%</td>
</tr>
<tr>
<td>10</td>
<td>4.16%</td>
<td>4.86%</td>
</tr>
</tbody>
</table>

### Table 559. Tolerance of the USART receiver when BRR[3:0] is different from 0000

<table>
<thead>
<tr>
<th>M bits</th>
<th>OVER8 bit = 0</th>
<th>OVER8 bit = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ONEBIT=0</td>
<td>ONEBIT=1</td>
</tr>
<tr>
<td>00</td>
<td>3.33%</td>
<td>3.88%</td>
</tr>
<tr>
<td>01</td>
<td>3.03%</td>
<td>3.53%</td>
</tr>
<tr>
<td>10</td>
<td>3.7%</td>
<td>4.31%</td>
</tr>
</tbody>
</table>
These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.
  
  In this case the USART measures the duration of the start bit (falling edge to rising edge).

- **Mode 1**: Any character starting with a 10xx bit pattern.
  
  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.

- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).
  
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.

- **Mode 3**: A 0x55 character frame.
  
  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

**Note:** The BRR value might be corrupted if the USART is disabled (UE=0) during an auto baud rate operation.
53.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:
- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in Figure 741.
Figure 741. Mute mode using Idle line detection

<table>
<thead>
<tr>
<th>RX</th>
<th>Data 1</th>
<th>Data 2</th>
<th>Data 3</th>
<th>Data 4</th>
<th>IDLE</th>
<th>Data 5</th>
<th>Data 6</th>
</tr>
</thead>
</table>

RWU | Mute mode | Normal mode |

MMRQ written to 1 | Idle frame detected

Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in Figure 742.
53.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.
53.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in Table 560.

Table 560. USART frame formats

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>USART frame(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>SB</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>SB</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>SB</td>
</tr>
</tbody>
</table>

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).
53.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to Section 53.4: USART implementation on page 2408.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in Section 53.5.5 has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value '1' are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE=1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL=1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit=1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN=0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN=1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a '1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown on the Figure 743: Break detection in LIN mode (11-bit break length - LBDL bit is set) on page 2435.

Examples of break frames are given on Figure 744: Break detection in LIN mode vs. Framing error detection on page 2436.
Figure 743. Break detection in LIN mode (11-bit break length - LBDL bit is set)

Case 1: break signal not long enough => break discarded, LBDF is not set

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td></td>
</tr>
<tr>
<td>Break state machine</td>
<td>Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 Idle</td>
</tr>
<tr>
<td>Read samples</td>
<td>0 0 0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

Case 2: break signal just long enough => break detected, LBDF is set

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td></td>
</tr>
<tr>
<td>Break state machine</td>
<td>Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 Idle</td>
</tr>
<tr>
<td>Read samples</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>LBDF</td>
<td>wait delimiter</td>
</tr>
</tbody>
</table>

Case 3: break signal long enough => break detected, LBDF is set

<table>
<thead>
<tr>
<th>RX line</th>
<th>Break frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture strobe</td>
<td></td>
</tr>
<tr>
<td>Break state machine</td>
<td>Idle Bit0 Bit1 Bit2 Bit3 Bit4 Bit5 Bit6 Bit7 Bit8 Bit9 Bit10 wait delimiter Idle</td>
</tr>
<tr>
<td>Read samples</td>
<td>0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>LBDF</td>
<td></td>
</tr>
</tbody>
</table>
**53.5.15 USART synchronous mode**

**Master mode**

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see Figure 745, Figure 746 and Figure 747).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).
Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE=1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 745. USART example of synchronous master transmission

Figure 746. USART data clock timing diagram in synchronous master mode (M bits =00)
Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

*Note:* When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (usart_ker_ck_pres) must be greater than 3 times the CK input frequency.

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see Figure 748).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.
Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- **Software NSS management (DIS_NSS = 1)**
  SPI slave is always selected and NSS input pin is ignored.
  The external NSS pin remains free for other application uses.

- **Hardware NSS management (DIS_NSS = 0)**
  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

**Note:** The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE=0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master.

The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

**SPI slave underrun error**

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.
In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 *usart_ker_ck* cycles before the first CK edge.

### 53.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

### 53.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register. The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.
53.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to Section 53.4: USART implementation on page 2408.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:
- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T=0 (character mode) and T=1 (block mode) are supported.

The USART must be configured as:
- 8 bits plus parity: M=1 and PCE=1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP=11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T=0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 749 shows examples of what can be seen on the data line with and without parity error.

![Figure 749. ISO 7816-3 asynchronous protocol](image)

When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.
- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.
The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guardtime). If the software wants to repeat it again, it must insure the minimum 2 baud periods required by the standard.

- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.

- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn’t repeat the character.

- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).

- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.

- The de-assertion of TC flag is unaffected by smartcard mode.

- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.

- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

**Note:** Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 750 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.
The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from \( \frac{\text{usart_ker_ck_pres}}{2} \) to \( \frac{\text{usart_ker_ck_pres}}{62} \), where \( \text{usart_ker_ck_pres} \) is the peripheral input clock divided by a programmed prescaler.

**Block mode (T = 1)**

In T=1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) - 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

**Note:** The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time -11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

**Note:** As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT - 11 or CWT - 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value.
(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

**Note:** The error checking code (LRC/CRC) must be computed/verified by software.

**Direct and inverse convention**

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=0, DATAINV=0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST=1, DATAINV=1.

**Note:** When logical data values are inverted (0=H, 1=L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.
Therefore, two methods are available for TS pattern recognition:

**Method 1**

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

**Method 2**

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen
(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

**53.5.19 USART IrDA SIR ENDEC block**

This section is relevant only when IrDA mode is supported. Refer to [Section 53.4: USART implementation on page 2408](#).

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies use of a Return to Zero, Inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see Figure 751).

The SIR Transmit encoder modulates the Non Return to Zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not
encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see Figure 752).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than 2 PSC periods (PSC is the prescaler value programmed in the USART_GTPR). Pulses of width less than 1 PSC period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than 2 periods are accepted as a pulse. The IrDA encoder/decoder doesn't work when PSC=0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to ‘1 stop bit’.

**IrDA low-power mode**

- **Transmitter**
  In low-power mode, the pulse width is not maintained at 3/16 of the bit period. Instead, the width of the pulse is 3 times the low-power baud rate which can be a minimum of 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < PSC< 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- **Receiver**
  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than 1/PSC. A valid low is accepted only if its duration is greater than 2 periods of the IrDA low-power baud clock (PSC value in the USART_GTPR).

**Note:**
A pulse of width less than two and greater than one PSC period(s) may or may not be rejected.

The receiver set up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception (IrDA is a half duplex protocol).
Figure 751. IrDA SIR ENDEC block diagram

Figure 752. IrDA data modulation (3/16) - normal mode
53.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

**Note:** Refer to Section 53.4: USART implementation on page 2408 to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 53.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to section Direct memory access controller (DMA)) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register.
5. Configure DMA interrupt generation after half/full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

**Note:** The DMAT bit must not be cleared before the DMA end of transfer.
Figure 753. Transmission using DMA

<table>
<thead>
<tr>
<th>TX line</th>
<th>TXE flag</th>
<th>DMA request</th>
<th>USART_TDR</th>
<th>TC flag</th>
<th>DMA writes USART_TDR</th>
<th>DMA TCIF flag (transfer complete)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle preamble</td>
<td>Set by hardware cleared by DMA read</td>
<td>Ignored by the DMA because the transfer is complete</td>
<td>F1</td>
<td>Cleared by software</td>
<td>Software waits until TC=1</td>
<td></td>
</tr>
<tr>
<td>Frame 1</td>
<td></td>
<td></td>
<td>F2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame 2</td>
<td></td>
<td></td>
<td>F3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frame 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (that is, TXFNF = 1).

**Reception using DMA**

DMA mode can be enabled for reception by setting the DMAR bit in USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to section Direct memory access controller (DMA)) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

**Note:** The DMAR bit must not be cleared before the DMA end of transfer.
Note: When FIFO management is enabled, the DMA request is triggered by Receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

53.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The Figure 755 shows how to connect two devices in this mode:

Figure 755. Hardware flow control between 2 USARTs
RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

**RS232 RTS flow control**

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 756* shows an example of communication with RTS flow control enabled.

*Figure 756. RS232 RTS flow control*

![Diagram of RS232 RTS flow control](image)

*Note:* When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

**RS232 CTS flow control**

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 757* shows an example of communication with CTS flow control enabled.
**Note:** For correct behavior, CTS must be deasserted at least 3 USART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

**RS485 driver enable**

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the USART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).
53.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (usart_wkup) if a specific action requiring the activation of the usart_pclk clock is needed:

- If FIFO mode is disabled
  usart_pclk clock has to be activated to empty the USART data register.
  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled
  usart_pclk clock has to be activated to:
  - to fill the TXFIFO
  - or to empty the RXFIFO
  In this case, the usart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFIE bit must be set before entering low-power mode.

This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific usart_wkup interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a usart_wkup interrupt is generated if the WUFIE bit is set.
Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.
Figure 758 shows the USART behavior when the wake-up event is verified.

**Figure 758. Wake-up event verified (wake-up event = address match, FIFO disabled)**

![Diagram showing USART behavior when wake-up event is verified](MSv40856V2)

Figure 759 shows the USART behavior when the wake-up event is not verified.

**Figure 759. Wake-up event not verified (wake-up event = address match, FIFO disabled)**

![Diagram showing USART behavior when wake-up event is not verified](MSv40857V2)

Note: The figures above are valid when address match or any received frame is used as wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.
Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see Section 53.5.9: Tolerance of the USART receiver to clock deviation).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to Table 558: Tolerance of the USART receiver when BRR [3:0] = 0000, the USART receiver tolerance equals 3.41%.

\[
D_{\text{WUmax}} = \frac{t_{\text{WUUSART}}}{11 \times T_{\text{bitmin}}} \\
T_{\text{bitmin}} = \frac{t_{\text{WUUSART}}}{11 \times D_{\text{WUmax}}}
\]

where \(t_{\text{WUUSART}}\) is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

\[
t_{\text{WUUSART}} = 3 \mu\text{s (values provided only as examples; for correct values, refer to the device datasheet).} \\
D_{\text{WUmax}} = \text{USART receiver tolerance} - \text{DREC} = 3.41\% - 1\% = 2.41\% \\
T_{\text{bitmin}} = \frac{3 \mu\text{s}}{11 \times 2.41\%} = 11.32 \mu\text{s}.
\]

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: \(1/11.32 \mu\text{s} = 88.36\text{ kbauds}\).

### 53.6 USART in low-power modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep(1)</td>
<td>No effect. USART interrupts cause the device to exit Sleep mode. The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode. The USART peripheral is powered down and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 53.4: USART implementation to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

### 53.7 USART interrupts

Refer to Table 562 for a detailed description of all USART interrupt requests.
<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART or UART</td>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>Write TDR</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFIFO full</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>Write TDR or write 1 in TXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>Write TDR</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>Write 1 in CTSCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>Write TDR or write 1 in TCCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Transmission Complete Before Guard Time</td>
<td>TCBGT</td>
<td>TCBGTE</td>
<td>Write TDR or write 1 in TCBGT</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 562. USART interrupt requests (continued)

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop(1) modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>USART or UART</strong></td>
<td>Receive data register not empty (data ready to be read)</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>Read RDR or write 1 in RXFRQ</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>Read RDR until RXFIFO empty or write 1 in RXFRQ</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive FIFO Full</td>
<td>RXFF(2)</td>
<td>RXFFIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>Read RDR</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RX-NEIE/RXFNEIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>Write 1 in IDLECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>Write 1 in PECE</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LIN break</td>
<td>LBDF</td>
<td>LBDFIE</td>
<td>Write 1 in LBDCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Noise error in multibuffer communication.</td>
<td>NE</td>
<td></td>
<td>Write 1 in NFCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun error in multibuffer communication.</td>
<td>ORE(3)</td>
<td>EIE</td>
<td>Write 1 in ORECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Framing Error in multibuffer communication.</td>
<td>FE</td>
<td></td>
<td>Write 1 in FECF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>Write 1 in CMCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Receiver timeout</td>
<td>RTOF</td>
<td>RTOFIE</td>
<td>Write 1 in RTOCCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>End of Block</td>
<td>EOBF</td>
<td>EOBIIE</td>
<td>Write 1 in EOBCE</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wake-up from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>Write 1 in WUC</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPI slave underrun error</td>
<td>UDR</td>
<td>EIE</td>
<td>Write 1 in UDRCF</td>
<td>No</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to Section 53.4: USART implementation for the list of supported Stop modes.

2. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

3. When OVRDIS = 0.
53.8 **USART registers**

Refer to Section 1.2 on page 120 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

53.8.1 **USART control register 1 (USART_CR1)**

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enable, FIFOEN = 1**

<table>
<thead>
<tr>
<th>Bit</th>
<th>RXFFIE</th>
<th>TXFFIE</th>
<th>FIFOEN</th>
<th>EOBIE</th>
<th>RTOIE</th>
<th>DEAT[4:0]</th>
<th>DEDT[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td>M1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>OVER8</td>
<td>CMIE</td>
<td>MME</td>
<td>M0</td>
<td>WAKE</td>
<td>PCE</td>
<td>PS</td>
</tr>
<tr>
<td>27</td>
<td>TXFNFIE</td>
<td>PEIE</td>
<td>TCIE</td>
<td>RXFNEIE</td>
<td>IDLEIE</td>
<td>TE</td>
<td>RE</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31** **RXFFIE**: RXFIFO Full interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when RXFF=1 in the USART_ISR register

**Bit 30** **TXFFIE**: TXFIFO empty interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when TXFE=1 in the USART_ISR register

**Bit 29** **FIFOEN**: FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE=0).

**Note**: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

**Bit 28** **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 start bit, 8 Data bits, n Stop bit

M[1:0] = 01: 1 start bit, 9 Data bits, n Stop bit

M[1:0] = 10: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE=0).

**Note**: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.
Bit 27  **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

- 0: Interrupt inhibited
- 1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

*Note:* If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 26  **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

- 0: Interrupt inhibited
- 1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

*Note:* If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 2408.

Bits 25:21  **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE=0).

*Note:* If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bits 20:16  **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE=0).

*Note:* If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 15  **OVER8**: Oversampling mode

- 0: Oversampling by 16
- 1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

*Note:* In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14  **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

- 0: Interrupt inhibited
- 1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13  **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

- 0: Receiver in active mode permanently
- 1: Receiver can switch between mute mode and active mode.

Bit 12  **MO**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the USART is disabled (UE=0).
Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE=0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7 **TXFNFIE**: TX FIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF =1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5 **RXFNEIE**: RX FIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE=1 or RXFNE=1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register
Bit 3  **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2  **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1  **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

**Note:** It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0  **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

**Note:** To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.
53.8.2 **USART control register 1 [alternate] (USART_CR1)**

Address offset: 0x00
Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled, FIFOEN = 0**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>FIFOEN</td>
<td>FIFO mode enable</td>
</tr>
<tr>
<td>29</td>
<td>EOBIE</td>
<td>End of block interrupt enable</td>
</tr>
<tr>
<td>28</td>
<td>M1</td>
<td>Word length</td>
</tr>
<tr>
<td>27</td>
<td>RTOIE</td>
<td>Receiver timeout interrupt enable</td>
</tr>
<tr>
<td>26</td>
<td>DEAT[4:0]</td>
<td>Data error interrupt mask</td>
</tr>
<tr>
<td>25</td>
<td>DEDT[4:0]</td>
<td>Data end timer mask</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Address offset</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

Bits 31:30: Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable

- 0: FIFO mode is disabled.
- 1: FIFO mode is enabled.

This bitfield can only be written when the USART is disabled (UE=0).

*Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.*

Bit 28 **M1**: Word length

- This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
- M[1:0] = 00: 1 start bit, 8 Data bits, n Stop bit
- M[1:0] = 01: 1 start bit, 9 Data bits, n Stop bit
- M[1:0] = 10: 1 start bit, 7 Data bits, n Stop bit

This bit can only be written when the USART is disabled (UE=0).

*Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bit 27 **EOBIE**: End of block interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

*Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bit 26 **RTOIE**: Receiver timeout interrupt enable

- This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

*Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 2408.*
Bits 25:21 **DEAT[4:0]:** Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE=0).

*Note:* If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bits 20:16 **DEDT[4:0]:** Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE=0).

*Note:* If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 15 **OVER8:** Oversampling mode

0: Oversampling by 16
1: Oversampling by 8

This bit can only be written when the USART is disabled (UE=0).

*Note:* In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE:** Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited
1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME:** Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently
1: Receiver can switch between mute mode and active mode.

Bit 12 **M0:** Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the USART is disabled (UE=0).

Bit 11 **WAKE:** Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line
1: Address mark

This bitfield can only be written when the USART is disabled (UE=0).

Bit 10 **PCE:** Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software.

Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled
1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE=0).
Bit 9  **PS**: Parity selection  
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.  
0: Even parity  
1: Odd parity  
This bitfield can only be written when the USART is disabled (UE=0).

Bit 8  **PEIE**: PE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever PE=1 in the USART_ISR register

Bit 7  **TXEIE**: Transmit data register empty  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TXE =1 in the USART_ISR register

Bit 6  **TCIE**: Transmission complete interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever TC=1 in the USART_ISR register

Bit 5  **RXNEIE**: Receive data register not empty  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever ORE=1 or RXNE=1 in the USART_ISR register

Bit 4  **IDLEIE**: IDLE interrupt enable  
This bit is set and cleared by software.  
0: Interrupt inhibited  
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3  **TE**: Transmitter enable  
This bit enables the transmitter. It is set and cleared by software.  
0: Transmitter is disabled  
1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.  
In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.
Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.
*Note*: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable
When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.
0: USART prescaler and outputs disabled, low-power mode
1: USART enabled
*Note*: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.
In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

### 53.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD[7:0]</td>
<td>RTOEN</td>
<td>ABRMOD[1:0]</td>
<td>ABREN</td>
<td>MSBFI</td>
<td>RST</td>
<td>DATAINV</td>
<td>TXINV</td>
<td>RXINV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SWAP</td>
<td>LINEN</td>
<td>STOP[1:0]</td>
<td>CLKEN</td>
<td>CPOL</td>
<td>CPHA</td>
<td>LBCL</td>
<td>Res.</td>
<td>LBDIE</td>
<td>LBDL</td>
<td>ADDM7</td>
<td>DIS_NSS</td>
<td>Res.</td>
<td>Res.</td>
<td>SLVEN</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:24 **ADD[7:0]:** Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:
- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

**Bit 23 RTOEN:** Receiver timeout enable

This bit is set and cleared by software.
0: Receiver timeout feature disabled.
1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

*Note:* If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bits 22:21 **ABRMOD[1:0]:** Auto baud rate mode

These bits are set and cleared by software.
00: Measurement of the start bit is used to detect the baud rate.
01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)
10: 0x7F frame detection.
11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE=0).

*Note:* If DATAINV=1 and/or MSBFIRST=1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

**Bit 20 ABREN:** Auto baud rate enable

This bit is set and cleared by software.
0: Auto baud rate detection is disabled.
1: Auto baud rate detection is enabled.

*Note:* If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

**Bit 19 MSBFIRST:** Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE=0).
Bit 18 **DATAINV**: Binary data inversion
   This bit is set and cleared by software.
   0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
   1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 17 **TXINV**: TX pin active level inversion
   This bit is set and cleared by software.
   0: TX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
   1: TX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
   This enables the use of an external inverter on the TX line.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 16 **RXINV**: RX pin active level inversion
   This bit is set and cleared by software.
   0: RX pin signal works using the standard logic levels (VDD =1/idle, Gnd=0/mark)
   1: RX pin signal values are inverted. ((VDD =0/mark, Gnd=1/idle).
   This enables the use of an external inverter on the RX line.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 15 **SWAP**: Swap TX/RX pins
   This bit is set and cleared by software.
   0: TX/RX pins are used as defined in standard pinout
   1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
   This bitfield can only be written when the USART is disabled (UE=0).

Bit 14 **LINEN**: LIN mode enable
   This bit is set and cleared by software.
   0: LIN mode disabled
   1: LIN mode enabled
   The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.
   This bitfield can only be written when the USART is disabled (UE=0).

   **Note**: *If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bits 13:12 **STOP[1:0]**: stop bits
   These bits are used for programming the stop bits.
   00: 1 stop bit
   01: 0.5 stop bit.
   10: 2 stop bits
   11: 1.5 stop bits
   This bitfield can only be written when the USART is disabled (UE=0).
Bit 11 **CLKEN**: Clock enable
This bit enables the user to enable the CK pin.
0: CK pin disabled
1: CK pin enabled

*Note*: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:

- UE = 0
- SCEN = 1
- GTPR configuration
- CLKEN = 1
- UE = 1

Bit 10 **CPOL**: Clock polarity
This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship.
0: Steady low value on CK pin outside transmission window
1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE=0).

*Note*: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 9 **CPHA**: Clock phase
This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see Figure 739 and Figure 740).
0: The first clock transition is the first data capture edge
1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE=0).

*Note*: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 8 **LBCL**: Last bit clock pulse
This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.
0: The clock pulse of the last data bit is not output to the CK pin
1: The clock pulse of the last data bit is output to the CK pin

*Caution*: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE=0).

*Note*: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable
Break interrupt mask (break detection using break delimiter).
0: Interrupt is inhibited
1: An interrupt is generated whenever LBDF=1 in the USART_ISR register

*Note*: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.
Bit 5  **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection
1: 11-bit break detection

This bit can only be written when the USART is disabled (UE=0).

*Note*: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 4  **ADDM7**: 7-bit Address Detection/4-bit Address Detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE=0)

*Note*: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3  **DIS_NSS**: When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.
1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note*: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bits 2:1  Reserved, must be kept at reset value.

Bit 0  **SLVEN**: Synchronous slave mode enable

When the SLVEN bit is set, the synchronous slave mode is enabled.

0: Slave mode disabled.
1: Slave mode enabled.

This bitfield can only be written when the USART is disabled (UE = 0).

*Note*: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

*Note*: The CPOL, CPHA and LBCL bits must not be written while the transmitter is enabled.

53.8.4  **USART control register 3 (USART_CR3)**

Address offset: 0x08
Reset value: 0x0000 0000

**FIFO mode enabled, FIFOEN = 1**
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration
000: TXFIFO reaches 1/8 of its depth
001: TXFIFO reaches 1/4 of its depth
010: TXFIFO reaches 1/2 of its depth
011: TXFIFO reaches 3/4 of its depth
100: TXFIFO reaches 7/8 of its depth
101: TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration
000: Receive FIFO reaches 1/8 of its depth
001: Receive FIFO reaches 1/4 of its depth
010: Receive FIFO reaches 1/2 of its depth
011: Receive FIFO reaches 3/4 of its depth
100: Receive FIFO reaches 7/8 of its depth
101: Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF=1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.
Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE=0).
**Note:** If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number of erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE=0).
When the USART is enabled (UE=1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
**Note:** If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the USART is disabled (UE=0).
**Note:** If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the USART is disabled (UE=0).
**Note:** If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 2408.

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE (case FIFO mode is enabled) before clearing the error flag.
This bit can only be written when the USART is disabled (UE=0).
**Note:** The reception errors are: parity error, framing error or noise error.
Bit 12 **OVRDIS**: Overrun Disable

- This bit is used to disable the receive overrun detection.
- 0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
- 1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.
- This bit can only be written when the USART is disabled (UE=0).

*Note: This control bit enables checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

- This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.
- 0: Three sample bit method
- 1: One sample bit method
- This bit can only be written when the USART is disabled (UE=0).

Bit 10 **CTSE**: CTS interrupt enable

- 0: Interrupt is inhibited
- 1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bit 9 **RTSE**: RTS enable

- 0: RTS hardware flow control disabled
- 1: RTS mode enabled, data is only transmitted when the RTS input is deasserted (tied to 0).
  - If the RTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while RTS is asserted, the transmission is postponed until RTS is deasserted.
- This bit can only be written when the USART is disabled (UE=0)

*Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bit 8 **DMAT**: DMA enable transmitter

- This bit is set/reset by software
- 1: DMA mode is enabled for transmission
- 0: DMA mode is disabled for transmission

Bit 7 **DMAR**: DMA enable receiver

- This bit is set/reset by software
- 1: DMA mode is enabled for reception
- 0: DMA mode is disabled for reception
Bit 5 **SCEN**: Smartcard mode enable
This bit is used for enabling smartcard mode.
0: Smartcard mode disabled
1: Smartcard mode enabled
This bitfield can only be written when the USART is disabled (UE=0).

*Note*: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 4 **NACK**: Smartcard NACK enable
0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bitfield can only be written when the USART is disabled (UE=0).

*Note*: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the USART is disabled (UE=0).

Bit 2 **IRLP**: IrDA low-power
This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode
This bit can only be written when the USART is disabled (UE=0).

*Note*: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 1 **IREN**: IrDA mode enable
This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled
This bit can only be written when the USART is disabled (UE=0).

*Note*: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1 or UDR = 1 in the USART_ISR register).
0: Interrupt inhibited
1: interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.
53.8.5  USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08
Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

| Bit 31:25 Reserved, must be kept at reset value. |
| Bit 24 | TCBGTIE: Transmission Complete before guard time, interrupt enable |
| 0: Interrupt inhibited |
| 1: USART interrupt generated whenever TCBGT=1 in the USART_ISR register |
| Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408. |
| Bit 23 Reserved, must be kept at reset value. |
| Bit 22 | WUFIE: Wake-up from low-power mode interrupt enable |
| 0: Interrupt inhibited |
| 1: USART interrupt generated whenever WUF=1 in the USART_ISR register |
| Note: WUFIE must be set before entering in low-power mode. |
| If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408. |
| Bits 21:20 WUS[1:0]: Wake-up from low-power mode interrupt flag selection |
| This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag). |
| 00: WUF active on address match (as defined by ADD[7:0] and ADDM7) |
| 01: Reserved. |
| 10: WUF active on start bit detection |
| 11: WUF active on RXNE/RXFNE. |
| This bitfield can only be written when the USART is disabled (UE=0). |
| Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408. |
Bits 19:17 **SCARCNT[2:0]:** Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE=0).

When the USART is enabled (UE=1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

**Note:** If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP:** Driver enable polarity selection

0: DE signal is active high.
1: DE signal is active low.

This bit can only be written when the USART is disabled (UE=0).

**Note:** If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 14 **DEM:** Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE=0).

**Note:** If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 53.4: USART implementation on page 2408.

Bit 13 **DDRE:** DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE/RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE=0).

**Note:** The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS:** Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE=0).

**Note:** This control bit enables checking the communication flow w/o reading the data.
Bit 11  **ONEBIT**: One sample bit method enable
This bit enables the user to select the sample method. When the one sample bit method is
selected the noise detection flag (NE) is disabled.
0: Three sample bit method
1: One sample bit method
This bit can only be written when the USART is disabled (UE=0).

Bit 10  **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF=1 in the USART_ISR register
*Note*: **If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.**

Bit 9  **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0).
If the CTS input is asserted while data is being transmitted, then the transmission completes
before stopping. If data is written into the data register while CTS is asserted, the
transmission is postponed until CTS is deasserted.
This bit can only be written when the USART is disabled (UE=0)
*Note*: **If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.**

Bit 8  **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The
transmission of data is expected to cease after the current character has been transmitted.
The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the USART is disabled (UE=0).
*Note*: **If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.**

Bit 7  **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6  **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bit 5  **SCEN**: Smartcard mode enable
This bit is used for enabling smartcard mode.
0: Smartcard mode disabled
1: Smartcard mode enabled
This bitfield can only be written when the USART is disabled (UE=0).
*Note*: **If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.**

Bit 4  **NACK**: Smartcard NACK enable
0: NACK transmission in case of parity error is disabled
1: NACK transmission during parity error is enabled
This bitfield can only be written when the USART is disabled (UE=0).
*Note*: **If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.**
Bit 3 HDSEL: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the USART is disabled (UE=0).

Bit 2 IRLP: IrDA low-power
This bit is used for selecting between normal and low-power IrDA modes
0: Normal mode
1: Low-power mode
This bit can only be written when the USART is disabled (UE=0).
Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value.
Refer to Section 53.4: USART implementation on page 2408.

Bit 1 IREN: IrDA mode enable
This bit is set and cleared by software.
0: IrDA disabled
1: IrDA enabled
This bit can only be written when the USART is disabled (UE=0).
Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value.
Refer to Section 53.4: USART implementation on page 2408.

Bit 0 EIE: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE=1 or ORE=1 or NE=1 or UDR = 1 in the USART_ISR register).
0: Interrupt inhibited
1: Interrupt generated when FE=1 or ORE=1 or NE=1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

53.8.6 USART baud rate register (USART_BRR)
This register can only be written when the USART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BRR[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **BRR[15:0]:** USART baud rate

**BRR[15:4]**


**BRR[3:0]**

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

### 53.8.7  USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>GT[7:0]</th>
<th>PSC[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

GT: Guard time register
PSC: Prescaler register

**rm**: read-modify-write
**rw**: read-write
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 GT[7:0]: Guard time value
   This bitfield is used to program the Guard time value in terms of number of baud clock
   periods.
   This is used in smartcard mode. The transmission complete flag is set after this guard time
   value.
   This bitfield can only be written when the USART is disabled (UE=0).
   Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value.
   Refer to Section 53.4: USART implementation on page 2408.

Bits 7:0 PSC[7:0]: Prescaler value
   Condition: IrDA low-power and normal IrDA mode
   PSC[7:0] = IrDA normal and Low-power baud rate
   This bitfield is used for programming the prescaler for dividing the USART source clock to
   achieve the low-power frequency:
   The source clock is divided by the value given in the register (8 significant bits):
   00000000: Reserved - do not program this value
   00000001: divides the source clock by 1
   00000010: divides the source clock by 2
   ...
   Condition: Smartcard mode
   PSC[4:0]: Prescaler value
   This bitfield is used for programming the prescaler for dividing the USART source clock to
   provide the smartcard clock.
   The value given in the register (5 significant bits) is multiplied by 2 to give the division factor
   of the source clock frequency:
   00000: Reserved - do not program this value
   00001: divides the source clock by 2
   00010: divides the source clock by 4
   00011: divides the source clock by 6
   ...
   This bitfield can only be written when the USART is disabled (UE=0).
   Note: Bits [7:5] must be kept cleared if smartcard mode is used.
   This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA
   modes are not supported. Refer to Section 53.4: USART implementation on page 2408.

53.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLEN[7:0]</td>
<td>RTO[23:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| RTO[15:0] |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bits 31:24 **BLEN[7:0]:** Block Length

This bitfield gives the block length in smartcard T=1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1-LEC/2-CRC) - 1.

Examples:

- BLEN = 0 -> 0 information characters + LEC
- BLEN = 1 -> 0 information characters + CRC
- BLEN = 255 -> 254 information characters + CRC (total 256 characters)

In smartcard mode, the block length counter is reset when TXE=0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE=0 (receiver disabled) and/or when the EOBCF bit is written to 1.

**Note:** This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]:** Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

**Note:** This value must only be programmed once per received character.

**Note:** RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to “0x00000000” when the Receiver timeout feature is not supported. Refer to Section 53.4: USART implementation on page 2408.

**53.8.9 USART request register (USART_RQR)**

Address offset: 0x18

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TXFRQ</th>
<th>RXFRQ</th>
<th>MMRQ</th>
<th>SBKRQ</th>
<th>ABRRQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

RM0477 Rev 6 2481/3791
Bits 31:5  Reserved, must be kept at reset value.

**Bit 4** TXFRQ: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFREQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

*Note:* In FIFO mode, the TXFNF flag is reset during the flush request until TxxFIF0 is empty in order to ensure that no data are written in the data register.

**Bit 3** RXFRQ: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE. This enables to discard the received data without reading them, and avoid an overrun condition.

**Bit 2** MMRQ: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

**Bit 1** SBKRQ: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

*Note:* When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

**Bit 0** ABRRQ: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

*Note:* If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

### 53.8.10 USART interrupt and status register (USART_ISR)

**Address offset:** 0x1C

**Reset value:** 0x0000 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

**FIFO mode enabled, FIFOEN = 1**

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXFNT</td>
<td>RXFNT</td>
<td>TCBGT</td>
<td>RXFF</td>
<td>TXFE</td>
<td>REACK</td>
<td>TEACK</td>
<td>WUF</td>
<td>RWU</td>
<td>SBKF</td>
<td>CMF</td>
<td>BUSY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABRF</td>
<td>ABRE</td>
<td>UDR</td>
<td>EOBF</td>
<td>RTOF</td>
<td>CTS</td>
<td>CTSIF</td>
<td>LBDW</td>
<td>TXFNF</td>
<td>TC</td>
<td>RXFNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:28 Reserved, must be kept at reset value.

Bit 27 TXFT: TXFIFO threshold flag
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG of USART_CR3 register, that is, the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFIE bit =1 (bit 31) in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 RXFT: RXFIFO threshold flag
This bit is set by hardware when the threshold programmed in RXFTCFG in USART_CR3 register is reached. This means that there are (RXFTCFG - 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFIE bit =1 (bit 27) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG threshold is configured to 101, RXFT flag is set if 16 data are available, that is, 15 data in the RXFIFO and 1 data in the USART_RDR.
Consequently, the 17th received data does not cause an overrun error. The overrun error occurs after receiving the 18th data.

Bit 25 TCBGT: Transmission complete before guard time flag
This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 53.4: USART implementation on page 2408.

Bit 24 RXFF: RXFIFO Full
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFIE bit =1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 TXFE: TXFIFO Empty
This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFIE bit =1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.
Bit 22 **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.
It can be used to verify that the USART is ready for reception before entering low-power mode.

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bit 21 **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register. An interrupt is generated if WUFIE=1 in the USART_CR3 register.

*Note: When UESM is cleared, WUF flag is also cleared.*
*If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bit 19 **RWU**: Receiver wake-up from mute mode
This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.
When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode
1: Receiver in mute mode

*Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.*

Bit 18 **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.
An interrupt is generated if CMIE=1 in the USART_CR1 register.

0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)
1: Reception ongoing
Bit 15 **ABRF**: Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE=1) (ABRE, RXFNE and FE are also set in this case).

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

*Note:* If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

*Note:* If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error
1: underrun error

*Note:* If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T=1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached
1: End of block (number of characters) reached

*Note:* If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE=1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached
1: Timeout value reached without any data reception

*Note:* If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.
Bit 10 **CTS**: CTS flag
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9 **CTSIF**: CTS interrupt flag
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.
An interrupt is generated if CTSIE=1 in the USART_CR3 register.
0: No change occurred on the CTS status line
1: A change occurred on the CTS status line

*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 **LBDF**: LIN break detection flag
This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.
An interrupt is generated if LBDIE = 1 in the USART_CR2 register.
0: LIN Break not detected
1: LIN break detected

*Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value.*

Refer to **Section 53.4: USART implementation on page 2408**.

Bit 7 **TXFNF**: TXFIFO not full
TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.
An interrupt is generated if the TXFNFIE bit =1 in the USART_CR1 register.
0: Transmit FIFO is full
1: Transmit FIFO is not full

*Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).
This bit is used during single buffer transmission.*

Bit 6 **TC**: Transmission complete
This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.
It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.
An interrupt is generated if TCIE = 1 in the USART_CR1 register.
The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.
0: Transmission has not completed
1: Transmission has completed

*Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.*
Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE=1 in the USART_CR1 register.

0: Data is not received
1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected
1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE=1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error
1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
Universal synchronous/asynchronous receiver transmitter (USART/UART) RM0477

53.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported
XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>ABRF ABRE UDR EOBF RTOF CTS CTSIF LBDF TXE TC RXNE IDLE ORE NE FE PE</td>
</tr>
<tr>
<td>r r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

Bit 2 NE: Noise detection flag
This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.
0: No noise is detected
1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 53.5.9: Tolerance of the USART receiver to clock deviation on page 2427).
This error is associated with the character in the USART_RDR.

Bit 1 FE: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 PE: Parity error
This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.
Bits 31:26  Reserved, must be kept at reset value.

Bit 25  **TCBGT**: Transmission complete before guard time flag
- This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
- It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE=1 in the USART_CR3 register.
- This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
- 0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)
- 1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

**Note:** If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 53.4: USART implementation on page 2408.

Bits 24:23  Reserved, must be kept at reset value.

Bit 22  **REACK**: Receive enable acknowledge flag
- This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.
- It can be used to verify that the USART is ready for reception before entering low-power mode.

**Note:** If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 21  **TEACK**: Transmit enable acknowledge flag
- This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.
- It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the USART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20  **WUF**: Wake-up from low-power mode flag
- This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.
- An interrupt is generated if WUFIE=1 in the USART_CR3 register.

**Note:** When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 19  **RWU**: Receiver wake-up from mute mode
- This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.
- When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.
- 0: Receiver in active mode
- 1: Receiver in mute mode

**Note:** If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.
<table>
<thead>
<tr>
<th>Bit 18</th>
<th>SBKF: Send break flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRE bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.</td>
<td></td>
</tr>
<tr>
<td>0: No break character transmitted</td>
<td></td>
</tr>
<tr>
<td>1: Break character transmitted</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 17</th>
<th>CMF: Character match flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.</td>
<td></td>
</tr>
<tr>
<td>An interrupt is generated if CMIE=1 in the USART_CR1 register.</td>
<td></td>
</tr>
<tr>
<td>0: No Character match detected</td>
<td></td>
</tr>
<tr>
<td>1: Character match detected</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 16</th>
<th>BUSY: Busy flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).</td>
<td></td>
</tr>
<tr>
<td>0: USART is idle (no reception)</td>
<td></td>
</tr>
<tr>
<td>1: Reception ongoing</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>ABRF: Auto baud rate flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE=1) (ABRE, RXNE and FE are also set in this case)</td>
<td></td>
</tr>
<tr>
<td>It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRQQ in the USART_IQR register.</td>
<td></td>
</tr>
<tr>
<td>Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 14</th>
<th>ABRE: Auto baud rate error</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)</td>
<td></td>
</tr>
<tr>
<td>It is cleared by software, by writing 1 to the ABRQQ bit in the USART_IQR register.</td>
<td></td>
</tr>
<tr>
<td>Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 13</th>
<th>UDR: SPI slave underrun error flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.</td>
<td></td>
</tr>
<tr>
<td>0: No underrun error</td>
<td></td>
</tr>
<tr>
<td>1: underrun error</td>
<td></td>
</tr>
<tr>
<td>Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
<th>EOBF: End of block flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bit is set by hardware when a complete block has been received (for example T=1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.</td>
<td></td>
</tr>
<tr>
<td>An interrupt is generated if EOBIE = 1 in the USART_CR1 register.</td>
<td></td>
</tr>
<tr>
<td>It is cleared by software, writing 1 to EOBCF in the USART_ICR register.</td>
<td></td>
</tr>
<tr>
<td>0: End of block not reached</td>
<td></td>
</tr>
<tr>
<td>1: End of block (number of characters) reached</td>
<td></td>
</tr>
<tr>
<td>Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 53.4: USART implementation on page 2408.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 11  **RTOF**: Receiver timeout
   This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.
   An interrupt is generated if RTOIE=1 in the USART_CR2 register.
   In smartcard mode, the timeout corresponds to the CWT or BWT timings.
   0: Timeout value not reached
   1: Timeout value reached without any data reception
   **Note:** If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.
   The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.
   If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10  **CTS**: CTS flag
   This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
   0: CTS line set
   1: CTS line reset
   **Note:** If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9  **CTSIF**: CTS interrupt flag
   This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.
   An interrupt is generated if CTSIE=1 in the USART_CR3 register.
   0: No change occurred on the CTS status line
   1: A change occurred on the CTS status line
   **Note:** If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8  **LBDF**: LIN break detection flag
   This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR register.
   An interrupt is generated if LBDIE = 1 in the USART_CR2 register.
   0: LIN Break not detected
   1: LIN break detected
   **Note:** If the USART does not support LIN mode, this bit is reserved and kept at reset value.
   Refer to Section 53.4: USART implementation on page 2408.

Bit 7  **TXE**: Transmit data register empty
   TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T=0 mode, in case of transmission failure).
   An interrupt is generated if the TXEIE bit =1 in the USART_CR1 register.
   0: Data register full
   1: Data register empty
Bit 6  **TC**: Transmission complete
  This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.
  An interrupt is generated if TCIE=1 in the USART_CR1 register.
  TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5  **RXNE**: Read data register not empty
  RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.
  An interrupt is generated if RXNEIE=1 in the USART_CR1 register.
  0: Data is not received
  1: Received data is ready to be read.

Bit 4  **IDLE**: Idle line detected
  This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.
  0: No Idle line is detected
  1: Idle line is detected
  **Note**: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).
  If mute mode is enabled (MME=1), IDLE is set if the USART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3  **ORE**: Overrun error
  This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE=1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.
  An interrupt is generated if RXNEIE=1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.
  1: Overrun error is detected
  **Note**: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.
  This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.
Bit 2 **NE**: Noise detection flag
This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.
0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to *Section 53.5.9: Tolerance of the USART receiver to clock deviation on page 2427*).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

*Note:* This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

*Note:* This error is associated with the character in the USART_RDR.

### 53.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>Res. Res. UDRCF EOBCF RTOCF Res. CTSCF LBDCF TCBGTCF TCCF TXFEC CF IDLECF ORECF NECF FECF PECF</td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w w</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag
Writing 1 to this bit clears the WUF flag in the USART_ISR register.

*Note:* If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to *Section 53.4: USART implementation on page 2408*.

Bits 19:18 Reserved, must be kept at reset value.
Bit 17 **CMCF**: Character match clear flag
  Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**: SPI slave underrun clear flag
  Writing 1 to this bit clears the UDRF flag in the USART_ISR register.
  
  *Note:* If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408

Bit 12 **EOBCF**: End of block clear flag
  Writing 1 to this bit clears the EOBF flag in the USART_ISR register.
  
  *Note:* If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 11 **RTOCF**: Receiver timeout clear flag
  Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
  
  *Note:* If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
  Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
  
  *Note:* If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 8 **LBDCF**: LIN break detection clear flag
  Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
  
  *Note:* If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
  Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
  Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
  Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
  Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
  Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
  Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
  Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
  Writing 1 to this bit clears the PE flag in the USART_ISR register.
### 53.8.13 USART receive data register (USART_RDR)

Address offset: 0x24  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:9</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 8:0</td>
<td><strong>RDR[8:0]</strong>: Receive data value</td>
</tr>
<tr>
<td></td>
<td>Contains the received data character.</td>
</tr>
<tr>
<td></td>
<td>The RDR register provides the parallel interface between the input shift register and the internal bus (see Section 53.5.1: USART block diagram).</td>
</tr>
<tr>
<td></td>
<td>When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.</td>
</tr>
</tbody>
</table>

### 53.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:9</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 8:0</td>
<td><strong>TDR[8:0]</strong>: Transmit data value</td>
</tr>
<tr>
<td></td>
<td>Contains the data character to be transmitted.</td>
</tr>
<tr>
<td></td>
<td>The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see Section 53.5.1: USART block diagram).</td>
</tr>
<tr>
<td></td>
<td>When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.</td>
</tr>
</tbody>
</table>

*Note: This register must be written only when TXE/TXFNF=1.*
53.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE=0).

Address offset: 0x2C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 3:0 PRESCALER[3:0]: Clock prescaler

The USART input clock can be divided by a prescaler factor:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 53.4: USART implementation on page 2408.
## 53.8.16 USART register map

### Table 563. USART register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>USART_CR1 FIFO mode enabled</td>
<td>0x1C0F0E</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x00</td>
<td>USART_CR1 FIFO mode disabled</td>
<td>0x0C0F0E</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x04</td>
<td>USART_CR2</td>
<td>0x180007</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x08</td>
<td>USART_CR3 FIFO mode enabled</td>
<td>0x081F0F20</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x08</td>
<td>USART_CR3 FIFO mode disabled</td>
<td>0x001F0F20</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0C</td>
<td>USART_BRR</td>
<td>0x0C0000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x10</td>
<td>USART_GTPR</td>
<td>0x140800</td>
<td>GT[7:0] PSC[7:0]</td>
</tr>
<tr>
<td>0x14</td>
<td>USART_RTOR</td>
<td>0x140C00</td>
<td>BLEN[7:0] RTO[23:0]</td>
</tr>
<tr>
<td>0x18</td>
<td>USART_RQR</td>
<td>0x180000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>USART_ISR FIFO mode enabled</td>
<td>0x1C0000</td>
<td>00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x1C</td>
<td>USART_ISR FIFO mode disabled</td>
<td>0x000000</td>
<td>00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 149 for the register boundary addresses.
54 Low-power universal asynchronous receiver transmitter (LPUART)

This section describes the low-power universal asynchronous receiver transmitter (LPUART).

54.1 Introduction

The LPUART is an UART which enables bidirectional UART communications with a limited power consumption. Only 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

54.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data
- Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
• Transfer detection flags:
  – Receive buffer full
  – Transmit buffer empty
  – Busy and end of transmission flags
• Parity control:
  – Transmits parity bit
  – Checks parity of received data byte
• Four error detection flags:
  – Overrun error
  – Noise detection
  – Frame error
  – Parity error
• Interrupt sources with flags
• Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
• Wake-up from Stop mode

54.3 LPUART implementation

The tables below describe LPUART implementation. It also includes USARTs and UARTs for comparison.

<table>
<thead>
<tr>
<th>Table 564. Instance implementation on STM32H7Rx/7Sx</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Instance</strong></td>
</tr>
<tr>
<td>USART1</td>
</tr>
<tr>
<td>USART2</td>
</tr>
<tr>
<td>USART3</td>
</tr>
<tr>
<td>UART4</td>
</tr>
<tr>
<td>UART5</td>
</tr>
<tr>
<td>UART7</td>
</tr>
<tr>
<td>UART8</td>
</tr>
<tr>
<td>LPUART1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 565. USART/LPUART features</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Modes/features</strong></td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>Hardware flow control for modem</td>
</tr>
<tr>
<td>Continuous communication using DMA</td>
</tr>
<tr>
<td>Multiprocessor communication</td>
</tr>
<tr>
<td>Synchronous mode (master/slave)</td>
</tr>
<tr>
<td>Smartcard mode</td>
</tr>
</tbody>
</table>
### Table 565. USART/LPUART features (continued)

<table>
<thead>
<tr>
<th>Modes/features&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>Full feature set</th>
<th>Basic feature set</th>
<th>Low-power feature set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-wire half-duplex communication</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>IrDA SIR ENDEC block</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>LIN mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Dual clock domain</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Receiver timeout interrupt</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Modbus communication</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Auto baud rate detection</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>Driver Enable</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>USART data length</td>
<td></td>
<td>7, 8 and 9 bits</td>
<td></td>
</tr>
<tr>
<td>Tx/Rx FIFO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tx/Rx FIFO size (bytes)</td>
<td>16</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
<td>X&lt;sup&gt;(2)&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

1. X = supported.
2. Wake-up supported from Stop mode.
54.4  LPUART functional description

54.4.1  LPUART block diagram

Figure 760. LPUART block diagram
54.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications
  LPUART bidirectional communications requires a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):
  - RX (Receive Data Input):
    RX is the serial data input.
  - TX (Transmit Data Output)
    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode
  The RS232 hardware flow control mode requires the following pins:
  - CTS (Clear To Send)
    When driven high, this signal blocks the data transmission at the end of the current transfer.
  - RTS (Request to send)
    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode
  The DE (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to Table 566 and Table 567 for the list of LPUART input/output pins and internal signals.

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPUART_RX</td>
<td>Input</td>
<td>Serial data receive input.</td>
</tr>
<tr>
<td>LPUART_TX</td>
<td>Output</td>
<td>Transmit data output.</td>
</tr>
<tr>
<td>LPUART_CTS</td>
<td>Input</td>
<td>Clear to send</td>
</tr>
<tr>
<td>LPUART_RTS</td>
<td>Output</td>
<td>Request to send</td>
</tr>
<tr>
<td>LPUART_DE(1)</td>
<td>Output</td>
<td>Driver enable</td>
</tr>
</tbody>
</table>

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 567. LPUART internal input/output signals

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lpuart_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
<tr>
<td>lpuart_ker_ck</td>
<td>Input</td>
<td>LPUART kernel clock</td>
</tr>
<tr>
<td>lpuart_wkup</td>
<td>Output</td>
<td>LPUART provides a wake-up interrupt</td>
</tr>
</tbody>
</table>
54.4.3 LPUART clocks

The simplified block diagram given in Section 54.4.1: LPUART block diagram shows two fully independent clock domains:

- **The **lpuart_pclk** clock domain
  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.

- **The **lpuart_ker_ck** kernel clock domain
  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck** is the same as the **lpuart_pclk** clock.

  There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**. The **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

54.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see Figure 734).

- 7-bit character length: M[1:0] = ‘10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

An **Idle character** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A **Break character** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.
Figure 761. LPUART word length programming

9-bit word length (M = 01), 1 Stop bit

8-bit word length (M = 00), 1 Stop bit

7-bit word length (M = 10), 1 Stop bit

** LBCL bit controls last data clock pulse
54.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bits wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through RXFTCFG and TXFTCFG bitfields in LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG bitfields.
  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG data have been received: 1 data in LPUART_RDR and (RXFTCFG - 1) data in the RXFIFO. As an example, when the RXFTCFG is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size - 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.

- The Tx interrupt is generated when the number of empty locations in the TXFIFO reaches the threshold programmed in the TXFTCFG bitfields.

54.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see Section 54.4.1: LPUART block diagram).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.
**Note:** The TE bit must be set before writing the data to be transmitted to the LPUART_TDR. The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

**Configurable stop bits**

The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13,12).

- **1 stop bit:** This is the default value of number of stop bits.
- **2 Stop bits:** This is supported by normal LPUART, single-wire and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

![Figure 762. Configurable stop bits](MS31885V1)

**Character transmission procedure**

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 54.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.
7. Write the data to send in the LPUART_TDR register. Repeat this operation for each
data to be transmitted in case of single buffer.
   - When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE
     flag.
   - When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to
     the TXFIFO. Write operations to the LPUART_TDR are performed when TXFNF flag
     is set. This flag remains set until the TXFIFO is full.

8. When the last data is written to the LPUART_TDR register, wait until TC=1. This
    indicates that the transmission of the last frame has completed.
   - When FIFO mode is disabled, this indicates that the transmission of the last frame
     has completed.
   - When FIFO mode is enabled, this indicates that both TXFIFO and shift register are
     empty.

   This check is required to avoid corrupting the last transmission when the LPUART is
   disabled or enters Halt mode.

**Single byte communication**

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by
  hardware to indicate that:
  - the data have been moved from the LPUART_TDR register to the shift register
    and data transmission has started;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the
    previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register
  stores the data in the TDR register, which is copied to the shift register at the end of the
  current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register
  places the data in the shift register, the data transmission starts, and the TXE bit is set.

- When FIFO mode is enabled, the TXFNF (TXFIFO Not Full) flag is set by hardware to
  indicate that:
  - the TXFIFO is not full;
  - the LPUART_TDR register is empty;
  - the next data can be written to the LPUART_TDR register without overwriting the
    previous data. When a transmission is ongoing, a write operation to the
LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXE is case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC=1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see Figure 763: TC/TXE behavior when transmitting).

![Figure 763. TC/TXE behavior when transmitting](image)

**Figure 763. TC/TXE behavior when transmitting**

**Note:** When FIFO management is enabled, the TXFNF flag is used for data transmission.

### Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see Figure 761).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.
When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

**Idle characters**

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

### 54.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

#### Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

#### Character reception

During an LPUART reception, data are shifted in least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

#### Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in [Section 54.4.13: Continuous communication using DMA and LPUART](#).
6. Set the RE bit LPUART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.
When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the Receive Data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

**Break character**

When a break character is received, the LPUART handles it as a framing error.

**Idle character**

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.
Overrun error

- **FIFO mode disabled**
  
  An overrun error occurs when a character is received when RXNE has not been reset. Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:
  
  - the ORE bit is set;
  - the RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed;
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXNEIE bit or EIE bit is set.

- **FIFO mode enabled**

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data can not be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:
  
  - the ORE bit is set;
  - the first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  - the shift register is overwritten. After that, any data received during overrun is lost.
  - an interrupt is generated if either the RXFNEIE bit or EIE bit is set.

  The ORE bit is reset by setting the ORECF bit in the ICR register.

*Note:* The **ORE bit, when set, indicates that at least 1 data has been lost.**

When the FIFO mode is disabled, there are two possibilities

- if RXNE=1, then the last valid data is stored in the receive register (RDR) and can be read,
- if RXNE=0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.

Selecting the clock source

The choice of the clock source is done through the Clock Control system (see Section Reset and clock controller (RCC)). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.
When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see Section Reset and clock controller (RCC)). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

**Figure 764. lpuart_ker_ck clock divider block diagram**

Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and Wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

*Note:* There is no noise detection for data.

**Framing error**

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:
- the FE bit is set by hardware;
- the invalid data is transferred from the Shift register to the LPUART_RDR register.
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.
Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

### 54.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

\[
\text{Tx/Rx baud} = \frac{256 \times \text{lpuart\_ker\_ck\_pres}}{\text{LPUARTDIV}}
\]

LPUARTDIV is defined in the LPUART_BRR register.

**Note:**

*The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence the baud rate register value must not be changed during communication.*

*It is forbidden to write values lower than 0x300 in the LPUART_BRR register.*

*The clock rate must range from 3 x baud rate to 4096 x baud rate.*

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock. For example, if the LPUART clock source frequency is 100 MHz, the maximum baud rate that can be reached is about 33 Mbauds.

<table>
<thead>
<tr>
<th>Table 568. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>S.No</strong></td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
</tbody>
</table>
Table 569. Error calculation for programmed baud rates at $f_{CK} = 100 \text{ MHz}$

<table>
<thead>
<tr>
<th>S.No</th>
<th>Desired</th>
<th>Actual</th>
<th>Value programmed in the baud rate register</th>
<th>$% \text{ Error} = \frac{\text{Calculated} - \text{Desired}}{\text{Desired B.rate}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>38400 bauds</td>
<td>38400,04 bauds</td>
<td>A2C2A</td>
<td>0,0001</td>
</tr>
<tr>
<td>2</td>
<td>57600 bauds</td>
<td>57600,06 bauds</td>
<td>6C81C</td>
<td>0,0001</td>
</tr>
<tr>
<td>3</td>
<td>115200 bauds</td>
<td>115200,12 bauds</td>
<td>3640E</td>
<td>0,0001</td>
</tr>
<tr>
<td>4</td>
<td>230400 bauds</td>
<td>230400,23 bauds</td>
<td>1B207</td>
<td>0,0001</td>
</tr>
<tr>
<td>5</td>
<td>460800 bauds</td>
<td>460804,61 bauds</td>
<td>D903</td>
<td>0,001</td>
</tr>
<tr>
<td>6</td>
<td>921600 bauds</td>
<td>921625,81 bauds</td>
<td>6C81</td>
<td>0,0028</td>
</tr>
<tr>
<td>7</td>
<td>4000 kbauds</td>
<td>4000000,00 bauds</td>
<td>1900</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>10000 kbauds</td>
<td>10000000,00 bauds</td>
<td>A00</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>20000 kbauds</td>
<td>20000000,00 bauds</td>
<td>500</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>30000 kbauds</td>
<td>33032258,06 bauds</td>
<td>307</td>
<td>0,1</td>
</tr>
</tbody>
</table>
54.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes which contribute to the total deviation are:

- **DTRA**: deviation due to the transmitter error (which also includes the deviation of the transmitter’s local oscillator)
- **DQUANT**: error due to the baud rate quantization of the receiver
- **DREC**: deviation of the receiver local oscillator
- **DTCL**: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

\[
\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{LPUART receiver tolerance}
\]

where

\[
\text{DWU} = \frac{t_{\text{WULPUART}}}{11 \times \text{Tbit}}
\]

when \( M[1:0] = 01 \):

\[
\text{DWU} = \frac{t_{\text{WULPUART}}}{11 \times \text{Tbit}}
\]

when \( M[1:0] = 00 \):

\[
\text{DWU} = \frac{t_{\text{WULPUART}}}{9 \times \text{Tbit}}
\]

\[
\text{DWU} = \frac{t_{\text{WULPUART}}}{10 \times \text{Tbit}}
\]

\[
\text{DWU} = \frac{t_{\text{WULPUART}}}{9 \times \text{Tbit}}
\]

\[
\text{WULPUART} \text{ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.}
\]

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in Table 570:

- Number of Stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

<table>
<thead>
<tr>
<th>M bits</th>
<th>768 &lt; BRR &lt; 1024</th>
<th>1024 &lt; BRR &lt; 2048</th>
<th>2048 &lt; BRR &lt; 4096</th>
<th>4096 ≤ BRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits (M=00), 1 Stop bit</td>
<td>1.82%</td>
<td>2.56%</td>
<td>3.90%</td>
<td>4.42%</td>
</tr>
<tr>
<td>9 bits (M=01), 1 Stop bit</td>
<td>1.69%</td>
<td>2.33%</td>
<td>2.53%</td>
<td>4.14%</td>
</tr>
<tr>
<td>7 bits (M=10), 1 Stop bit</td>
<td>2.08%</td>
<td>2.86%</td>
<td>4.35%</td>
<td>4.42%</td>
</tr>
<tr>
<td>8 bits (M=00), 2 Stop bit</td>
<td>2.08%</td>
<td>2.86%</td>
<td>4.35%</td>
<td>4.42%</td>
</tr>
<tr>
<td>9 bits (M=01), 2 Stop bit</td>
<td>1.82%</td>
<td>2.56%</td>
<td>3.90%</td>
<td>4.42%</td>
</tr>
<tr>
<td>7 bits (M=10), 2 Stop bit</td>
<td>2.34%</td>
<td>3.23%</td>
<td>4.92%</td>
<td>4.42%</td>
</tr>
</tbody>
</table>
Note: The data specified in Table 570 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when $M$ bits = 00 (11-bit times when $M=01$ or 9-bit times when $M = 10$).

54.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all non addressed receivers.

The non addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two \text{lpuart\_ker\_ck} cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE=0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in Figure 765.
Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE=1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in Figure 766.
54.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in Table 571.

### Table 571: LPUART frame formats

<table>
<thead>
<tr>
<th>M bits</th>
<th>PCE bit</th>
<th>LPUART frame&lt;sup&gt;(1)&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>SB</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>SB</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>SB</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>SB</td>
</tr>
</tbody>
</table>

2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data=00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).
Parity checking in reception
If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission
If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS=0) or an odd number of “1s” if odd parity is selected (PS=1)).

54.4.12 LPUART single-wire half-duplex communication
Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and Full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:
• The TX and RX lines are internally connected.
• The RX pin is no longer used
• The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

54.4.13 Continuous communication using DMA and LPUART
The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 54.3: LPUART implementation on page 2500 to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 54.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA
DMA mode can be enabled for transmission by setting DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section Direct memory access controller) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.

2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.

3. Configure the total number of bytes to be transferred to the DMA control register.

4. Configure the channel priority in the DMA register.

5. Configure DMA interrupt generation after half/full transfer as required by the application.

6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.

7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (the TCIF flag is set in the DMA_ISR register), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC=1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

**Note:** The DMAT bit must not be cleared before the DMA end of transfer.

**Figure 767. Transmission using DMA**

**Note:** When FIFO management is enabled, the DMA request is triggered by Transmit FIFO not full (that is, TXFNF = 1).
Reception using DMA

DMA mode can be enabled for reception by setting the DMAR bit in LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section Direct memory access controller (DMA)) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register.
5. Configure interrupt generation after half/full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA Controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 768. Reception using DMA

Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt.
enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

54.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. Figure 769 shows how to connect two devices in this mode.

Figure 769. Hardware flow control between two LPUARTs

RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE=1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. Figure 770 shows an example of communication with RTS flow control enabled.

Figure 770. RS232 RTS flow control

Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.
RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE=0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. Figure 771 shows an example of communication with CTS flow control enabled.

**Figure 771. RS232 CTS flow control**

![Diagram of RS232 CTS flow control]

Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The de-assertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.
The LPUART DEAT and DEDT are expressed in LPUART clock source (fCK) cycles:

- The Driver enable assertion time equals
  - \((1 + (DEAT \times P)) \times f_{CK}\), if \(P \neq 0\)
  - \((1 + DEAT) \times f_{CK}\), if \(P = 0\)
- The Driver enable de-assertion time equals
  - \((1 + (DEDT \times P)) \times f_{CK}\), if \(P \neq 0\)
  - \((1 + DEDT) \times f_{CK}\), if \(P = 0\)

where \(P = BRR[20:11]\)

### 54.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.

The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set. When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (lpuart_wkup) if a specific action requiring the activation of the lpuart_pclk clock is needed:

- If FIFO mode is disabled
  lpuart_pclk clock has to be activated to empty the LPUART data register.
  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled
  lpuart_pclk clock has to be activated
  - to fill the TXFIFO
  - or to empty the RXFIFO
  In this case, the lpuart_wkup interrupt source can be:
    - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
    - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
    - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.
  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:

- TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
- RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.
Alternatively, a specific lpuart_wkup interrupt may be selected through the WUS bitfields. When the wake-up event is detected, the WUF flag is set by hardware and lpuart_wkup interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and re-enabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not waken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 772 shows the behavior when the wake-up event is verified.
Figure 772. Wake-up event verified (wake-up event = address match, FIFO disabled)

Figure 773 shows the behavior when the wake-up event is not verified.

Figure 773. Wake-up event not verified (wake-up event = address match, FIFO disabled)

Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.
Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see Section 54.4.9: Tolerance of the LPUART receiver to clock deviation).

Let us take the example of $\text{OVER8} = 0$, $\text{M bits} = 01$, $\text{ONEBIT} = 0$ and $\text{BRR [3:0]} = 0000$.

In these conditions, according to Table 570: Tolerance of the LPUART receiver, the LPUART receiver tolerance equals 3.41%.

$$\text{DTRA} + \text{DQUANT} + \text{DREC} + \text{DTCL} + \text{DWU} < \text{LPUART receiver tolerance}$$

$$\text{DWU}_{\text{max}} = \frac{t_{\text{WULPUART}}}{11 \times T_{\text{bit Min}}}$$

$$T_{\text{bit Min}} = \frac{t_{\text{WULPUART}}}{11 \times \text{DWU}_{\text{max}}}$$

where $t_{\text{WULPUART}}$ is the wake-up time from low-power mode.

If we consider the ideal case where $\text{DTRA}$, $\text{DQUANT}$, $\text{DREC}$ and $\text{DTCL}$ parameters are at 0%, the maximum value of $\text{DWU}$ is 3.41%. In reality, we need to consider at least the $\text{lpuart}_{\text{ker ck}}$ inaccuracy.

For example, if HSI is used as $\text{lpuart}_{\text{ker ck}}$, and the HSI inaccuracy is of 1%, then we obtain:

$$t_{\text{WULPUART}} = 3 \; \mu\text{s}$$

(values provided only as examples; for correct values, refer to the device datasheet).

$$\text{D}_{\text{WU}}_{\text{max}} = 3.41% - 1% = 2.41%$$

$$T_{\text{bit min}} = 3 \; \mu\text{s} / (11 \times 2.41%) = 11.32 \; \mu\text{s}.$$

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: $1/11.32 \; \mu\text{s} = 88.36 \; \text{kbauds}.

54.5 LPUART in low-power modes

Table 572. Effect of low-power modes on the LPUART

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. LPUART interrupts cause the device to exit Sleep mode.</td>
</tr>
</tbody>
</table>
| Stop$^{(1)}$ | The content of the LPUART registers is kept.  
The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode. |
| Standby | The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode. |

1. Refer to Section 54.3: LPUART implementation to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.
54.6 **LPUART interrupts**

Refer to *Table 573* for a detailed description of all LPUART interrupt requests.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
<th>Exit from Stop modes</th>
<th>Exit from Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmit data register empty</td>
<td>TXE</td>
<td>TXEIE</td>
<td>Write TDR</td>
<td></td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Transmit FIFO Not Full</td>
<td>TXFNF</td>
<td>TXFNFIE</td>
<td>TXFIFO full</td>
<td></td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Transmit FIFO Empty</td>
<td>TXFE</td>
<td>TXFEIE</td>
<td>Write TDR or write 1 in TXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transmit FIFO threshold reached</td>
<td>TXFT</td>
<td>TXFTIE</td>
<td>Write TDR</td>
<td></td>
<td>Yes</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>CTS interrupt</td>
<td>CTSIF</td>
<td>CTSIE</td>
<td>Write 1 in CTSCF</td>
<td></td>
<td>No</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>Transmission Complete</td>
<td>TC</td>
<td>TCIE</td>
<td>Write TDR or write 1 in TCCF</td>
<td>No</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive data register not empty</td>
<td>RXNE</td>
<td>RXNEIE</td>
<td>Read RDR or write 1 in RXFRQ</td>
<td>Yes</td>
<td>No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO Not Empty</td>
<td>RXFNE</td>
<td>RXFNEIE</td>
<td>Read RDR until RXFIFO empty or write 1 in RXFRQ</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive FIFO Full</td>
<td>RXFF(2)</td>
<td>RXFFIE</td>
<td>Read RDR</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Receive FIFO threshold reached</td>
<td>RXFT</td>
<td>RXFTIE</td>
<td>Read RDR</td>
<td></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Overrun error detected</td>
<td>ORE</td>
<td>RX-NEIE/RX FNEIE</td>
<td>Write 1 in ORECF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Idle line detected</td>
<td>IDLE</td>
<td>IDLEIE</td>
<td>Write 1 in IDLECF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Parity error</td>
<td>PE</td>
<td>PEIE</td>
<td>Write 1 in PECF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Noise error in multibuffer communication</td>
<td>NE</td>
<td></td>
<td>Write 1 in NFCF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Overrun error in multibuffer communication</td>
<td>ORE(3)</td>
<td>EIE</td>
<td>Write 1 in ORECF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Framing Error in multibuffer communication</td>
<td>FE</td>
<td></td>
<td>Write 1 in FECF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Character match</td>
<td>CMF</td>
<td>CMIE</td>
<td>Write 1 in CMCF</td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Wake-up from low-power mode</td>
<td>WUF</td>
<td>WUFIE</td>
<td>Write 1 in WUC</td>
<td></td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>
1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to Section 54.3: LPUART implementation for the list of supported Stop modes.

2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).

3. When OVRDIS = 0.

54.7 LPUART registers

Refer to Section 1.2 on page 120 for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

54.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXFFIE</td>
<td>TXFEIE</td>
<td>FIFOEN</td>
<td>M1</td>
<td>Res</td>
<td>Res</td>
<td>DEAT[4:0]</td>
<td>DEDT[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>RXFFIE: RXFIFO Full interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: An LPUART interrupt is generated when RXFF=1 in the LPUART_ISR register</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>TXFEIE: TXFIFO empty interrupt enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Interrupt is inhibited</td>
<td></td>
</tr>
<tr>
<td>1: An LPUART interrupt is generated when TXFE=1 in the LPUART_ISR register</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>FIFOEN: FIFO mode enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: FIFO mode is disabled</td>
<td></td>
</tr>
<tr>
<td>1: FIFO mode is enabled</td>
<td></td>
</tr>
</tbody>
</table>
Bit 28  **M1**: Word length  
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
- \( M[1:0] = 00 \): 1 Start bit, 8 Data bits, \( n \) Stop bit
- \( M[1:0] = 01 \): 1 Start bit, 9 Data bits, \( n \) Stop bit
- \( M[1:0] = '10 \): 1 Start bit, 7 Data bits, \( n \) Stop bit
This bit can only be written when the LPUART is disabled (UE=0).

*Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:21  **DEAT[4:0]**: Driver Enable assertion time  
This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in `lpuart_ker_ck` clock cycles. For more details, refer to Section 53.5.21: RS232 hardware flow control and RS485 driver enable. This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16  **DEDT[4:0]**: Driver Enable deassertion time  
This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in `lpuart_ker_ck` clock cycles. For more details, refer to Section 54.4.14: RS232 hardware flow control and RS485 driver enable.  
If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed. This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  Reserved, must be kept at reset value.

Bit 14  **CMIE**: Character match interrupt enable  
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13  **MME**: Mute mode enable  
This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.
- 0: Receiver in active mode permanently
- 1: Receiver can switch between mute mode and active mode.

Bit 12  **M0**: Word length  
This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description). This bit can only be written when the LPUART is disabled (UE=0).

Bit 11  **WAKE**: Receiver wake-up method  
This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.
- 0: Idle line
- 1: Address mark
This bitfield can only be written when the LPUART is disabled (UE=0).
Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever TXFNF =1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever ORE=1 or RXFNE=1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

**Note:** During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.
In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.
Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.
When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode.

*Note*: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

*Note*: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

### 54.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00
Reset value: 0x0000 0000
The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled, FIFOEN = 0**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res.</td>
<td>CMIE</td>
<td>MME</td>
<td>M0</td>
<td>WAKE</td>
<td>PCE</td>
<td>PS</td>
<td>PEIE</td>
<td>TXIEIE</td>
<td>TCIE</td>
<td>RXNEIE</td>
<td>IDLEIE</td>
<td>TE</td>
<td>RE</td>
<td>UESM</td>
<td>UE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **FIFOEN**: FIFO mode enable
        This bit is set and cleared by software.
        0: FIFO mode is disabled.
        1: FIFO mode is enabled.

Bit 28  **M1**: Word length
        This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
        \[ M[1:0] = 00: 1 \text{ Start bit, 8 Data bits, n Stop bit} \]
        \[ M[1:0] = 01: 1 \text{ Start bit, 9 Data bits, n Stop bit} \]
        \[ M[1:0] = '10: 1 \text{ Start bit, 7 Data bits, n Stop bit} \]
        This bit can only be written when the LPUART is disabled (UE=0).

*Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.*

Bits 27:26  Reserved, must be kept at reset value.

Bits 25:21  **DEAT[4:0]**: Driver Enable assertion time
        This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer [Section 53.5.21: RS232 hardware flow control and RS485 driver enable](#).
        This bitfield can only be written when the LPUART is disabled (UE=0).

Bits 20:16  **DEDT[4:0]**: Driver Enable deassertion time
        This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in lpuart_ker_ck clock cycles. For more details, refer [Section 54.4.14: RS232 hardware flow control and RS485 driver enable](#).
        If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.
        This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  Reserved, must be kept at reset value.

Bit 14  **CMIE**: Character match interrupt enable
        This bit is set and cleared by software.
        0: Interrupt is inhibited
        1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13  **MME**: Mute mode enable
        This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.
        0: Receiver in active mode permanently
        1: Receiver can switch between mute mode and active mode.

Bit 12  **M0**: Word length
        This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).
        This bit can only be written when the LPUART is disabled (UE=0).
Bit 11 WAKE: Receiver wake-up method
This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 10 PCE: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M=1; 8th bit if M=0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 9 PS: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 8 PEIE: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever PE=1 in the LPUART_ISR register

Bit 7 TXEIE: Transmit data register empty
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever TXE =1 in the LPUART_ISR register

Bit 6 TCIE: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever TC=1 in the LPUART_ISR register

Bit 5 RXNEIE: Receive data register not empty
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated whenever ORE=1 or RXNE=1 in the LPUART_ISR register

Bit 4 IDLEIE: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated whenever IDLE=1 in the LPUART_ISR register
Bit 3 TE: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled

*Note:* During transmission, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 RE: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 UESM: LPUART enable in low-power mode
When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode. When this bit is set, the LPUART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: LPUART not able to wake up the MCU from low-power mode.
1: LPUART able to wake up the MCU from low-power mode.

*Note:* It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 UE: LPUART enable
When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.
0: LPUART prescaler and outputs disabled, low-power mode
1: LPUART enabled

*Note:* To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.
The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

### 54.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04
Reset value: 0x00000000

<table>
<thead>
<tr>
<th>Field</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>MSBF1</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>DATAINV</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>TXINV</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWAP</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>ADDM7</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

2536/3791  RM0477 Rev 6
Bits 31:24  **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter should be equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.

- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter should be equal to 1.

- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the LPUART is disabled (UE = 0).

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

Bit 19  **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

- 0: data is transmitted/received with data bit 0 first, following the start bit.
- 1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 18  **DATAINV**: Binary data inversion

This bit is set and cleared by software.

- 0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
- 1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 17  **TXINV**: TX pin active level inversion

This bit is set and cleared by software.

- 0: TX pin signal works using the standard logic levels ($V_{DD} =1$/idle, Gnd=0/mark)
- 1: TX pin signal values are inverted. (($V_{DD} =0$/mark, Gnd=1/idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 16  **RXINV**: RX pin active level inversion

This bit is set and cleared by software.

- 0: RX pin signal works using the standard logic levels ($V_{DD} =1$/idle, Gnd=0/mark)
- 1: RX pin signal values are inverted. (($V_{DD} =0$/mark, Gnd=1/idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 15  **SWAP**: Swap TX/RX pins

This bit is set and cleared by software.

- 0: TX/RX pins are used as defined in standard pinout
- 1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the LPUART is disabled (UE=0).

Bit 14  Reserved, must be kept at reset value.
54.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

**FIFO mode enabled, FIFOEN = 1**

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXFTCFG[2:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>RXFTI E</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>TXFTE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>WUFE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>WUS1</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>WUS0</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Res</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Res</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>CTSIE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>CTSE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>RTSE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>DMAT</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>DMAR</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Res</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>HDSEL</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>Res</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>EIE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:29 **TXFTCFG[2:0]**: TXFIFO threshold configuration
- 000: TXFIFO reaches 1/8 of its depth.
- 001: TXFIFO reaches 1/4 of its depth.
- 110: TXFIFO reaches 1/2 of its depth.
- 100: TXFIFO reaches 7/8 of its depth.
- 101: TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG.

Bits 27:25 **RXFTCFG[2:0]**: Receive FIFO threshold configuration
- 000: Receive FIFO reaches 1/8 of its depth.
- 001: Receive FIFO reaches 1/4 of its depth.
- 110: Receive FIFO reaches 1/2 of its depth.
- 011: Receive FIFO reaches 3/4 of its depth.
- 100: Receive FIFO reaches 7/8 of its depth.
- 101: Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
- 0: Interrupt is inhibited
- 1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
- 0: Interrupt inhibited
- 1: LPUART interrupt generated whenever WUF=1 in the LPUART_ISR register

**Note**: *WUFIE must be set before entering in low-power mode. If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wake-up from low-power mode flag).
- 00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
- 01: Reserved.
- 10: WUF active on start bit detection
- 11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE=0).

**Note**: *If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.*
Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
- 0: DE signal is active high.
- 1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
- 0: DE function is disabled.
- 1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13 **DDRE**: DMA Disable on reception Error
- 0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
- 1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).

*Note: The reception errors are: parity error, framing error or noise error.*

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
- 0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
- 1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).

*Note: This control bit enables checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
- 0: Interrupt is inhibited
- 1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
- 0: CTS hardware flow control disabled
- 1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0).
If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable
- 0: RTS hardware flow control disabled
- 1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE=0).
Low-power universal asynchronous receiver transmitter (LPUART)

Bit 7 DMAT: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 DMAR: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 HDSEL: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE=0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 EIE: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.

54.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08
Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0
Bits 31:23  Reserved, must be kept at reset value.

Bit 22  **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF=1 in the LPUART_ISR register

*Note: WUFIE must be set before entering in low-power mode.*
*If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.*

Bits 21:20  **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (Wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE=0).

*Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.*

Bits 19:16  Reserved, must be kept at reset value.

Bit 15  **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 14  **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE=0).

Bit 13  **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE=0).

*Note: The reception errors are: parity error, framing error or noise error.*

Bit 12  **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE=0).

*Note: This control bit enables checking the communication flow w/o reading the data.*
Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

   0: Interrupt is inhibited
   1: An interrupt is generated whenever CTSIF=1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

   0: CTS hardware flow control disabled
   1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0).
   If the CTS input is asserted while data is being transmitted, then the transmission completes
   before stopping. If data is written into the data register while CTS is asserted, the
   transmission is postponed until CTS is deasserted.
   This bit can only be written when the LPUART is disabled (UE=0)

Bit 8 **RTSE**: RTS enable

   0: RTS hardware flow control disabled
   1: RTS output enabled, data is only requested when there is space in the receive buffer. The
   transmission of data is expected to cease after the current character has been transmitted.
   The RTS output is deasserted (pulled to 0) when data can be received.
   This bit can only be written when the LPUART is disabled (UE=0).

Bit 7 **DMAT**: DMA enable transmitter

   This bit is set/reset by software
   1: DMA mode is enabled for transmission
   0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

   This bit is set/reset by software
   1: DMA mode is enabled for reception
   0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection

   Selection of single-wire half-duplex mode
   0: Half-duplex mode is not selected
   1: Half-duplex mode is selected
   This bit can only be written when the LPUART is disabled (UE=0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable

   Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing
   error, overrun error or noise flag (FE=1 or ORE=1 or NE=1 in the LPUART_ISR register).
   0: Interrupt is inhibited
   1: An interrupt is generated when FE=1 or ORE=1 or NE=1 in the LPUART_ISR register.
54.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE=0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRR[19:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

BRR[15:0]

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 BRR[19:0]: LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.
Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high fck values. fck must be in the range [3 x baud rate..4096 x baud rate].

54.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXFRQ</td>
<td>RXFRQ</td>
<td>MMRQ</td>
<td>SBKRQ</td>
<td>Res.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

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

Bit 4 TXFRQ: Transmit data flush request
This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 RXFRQ: Receive data flush request
Writing 1 to this bit clears the RXNE flag.
This enables discarding the received data without reading it, and avoid an overrun condition.
Bit 2 MMRQ: Mute mode request
Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 SBKRQ: Send break request
Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

54.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C
Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TXFT</td>
<td>RXFT</td>
<td>RXFF</td>
<td>TXFE</td>
<td>REACK</td>
<td>TEACK</td>
<td>WUF</td>
<td>RWU</td>
<td>SBKF</td>
<td>CMF</td>
<td>BUSY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTS</td>
<td>CTSIF</td>
<td>TXFNF</td>
<td>TC</td>
<td>RXFNE</td>
<td>IDLE</td>
<td>ORE</td>
<td>NE</td>
<td>FE</td>
<td>PE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 TXFT: TXFIFO threshold flag
This bit is set by hardware when the TXFIFO reaches the threshold programmed in TXFTCFG in LPUART_CR3 register, that is, the TXFIFO contains TXFTCFG empty locations. An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 RXFT: RXFIFO threshold flag
This bit is set by hardware when the RXFIFO reaches the threshold programmed in RXFTCFG in LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG data. An interrupt is generated if the RXFTIE bit = 1 (bit 27) in the LPUART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Bit 25 Reserved, must be kept at reset value.

Bit 24 RXFF: RXFIFO Full
This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register).
An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.
0: RXFIFO is not Full.
1: RXFIFO is Full.
Bit 23 **TXFE**: TXFIFO Empty
This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.
An interrupt is generated if the TXFEIE bit =1 (bit 30) in the LPUART_CR1 register.
0: TXFIFO is not empty.
1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.
It can be used to verify that the LPUART is ready for reception before entering low-power mode.
*Note*: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.
*Note*: When UESM is cleared, WUF flag is also cleared.
*If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.

Bit 19 **RWU**: Receiver wake-up from mute mode
This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode
*Note*: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE=1 in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected
Bit 16  **BUSY**: Busy flag  
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)  
1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10  **CTS**: CTS flag  
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set  
1: CTS line reset  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 9  **CTSIF**: CTS interrupt flag  
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register. An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.
0: No change occurred on the CTS status line  
1: A change occurred on the CTS status line  
*Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.*

Bit 8 Reserved, must be kept at reset value.

Bit 7  **TXFNF**: TXFIFO not full  
TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.
The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time). An interrupt is generated if the TXFNFIE bit =1 in the LPUART_CR1 register.
0: Data register is full/Transmit FIFO is full.  
1: Data register/Transmit FIFO is not full.  
*Note: This bit is used during single buffer transmission.*

Bit 6  **TC**: Transmission complete  
This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set. An interrupt is generated if TCIE = 1 in the LPUART_CR1 register. The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.
0: Transmission has not completed  
1: Transmission has completed  
*Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.*
Bit 5 **RXFNE**: RXFIFO not empty
RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.
The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.
An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register.
0: Data is not received
1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected
This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.
0: No Idle line is detected
1: Idle line is detected

*Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).*

If mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.

Bit 3 **ORE**: Overrun error
This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.
An interrupt is generated if RXFNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.
1: Overrun error is detected

*Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.*

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.
Bit 2 **NE**: Start bit noise detection flag
This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.
0: No noise is detected
1: Noise is detected

*Note:* This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

*Note:* This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error
This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

*Note:* This error is associated with the character in the LPUART_RDR.

54.7.9 **LPUART interrupt and status register [alternate] (LPUART_ISR)**
Address offset: 0x1C
Reset value: 0x0000 00C0
The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

**FIFO mode disabled, FIFOEN = 0**
Bits 31:23  Reserved, must be kept at reset value.

Bit 22  **REACK**: Receive enable acknowledge flag
This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.
It can be used to verify that the LPUART is ready for reception before entering low-power mode.

*Note:* If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21  **TEACK**: Transmit enable acknowledge flag
This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.
It can be used when an idle frame request is generated by writing TE=0, followed by TE=1 in the LPUART_CR1 register, in order to respect the TE=0 minimum period.

Bit 20  **WUF**: Wake-up from low-power mode flag
This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE=1 in the LPUART_CR3 register.

*Note:* When UESM is cleared, WUF flag is also cleared.
If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.

Bit 19  **RWU**: Receiver wake-up from mute mode
This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode

*Note:* If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18  **SBKF**: Send break flag
This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17  **CMF**: Character match flag
This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE=1 in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16  **BUSY**: Busy flag
This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: Reception ongoing

Bits 15:11  Reserved, must be kept at reset value.
Bit 10 **CTS**: CTS flag  
This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.  
0: CTS line set  
1: CTS line reset  
*Note*: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag  
This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.  
An interrupt is generated if CTSIE=1 in the LPUART_CR3 register.  
0: No change occurred on the CTS status line  
1: A change occurred on the CTS status line  
*Note*: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE**: Transmit data register empty  
TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.  
An interrupt is generated if the TXEIE bit =1 in the LPUART_CR1 register.  
0: Data register full  
1: Data register empty  
*Note*: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete  
This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.  
An interrupt is generated if TCIE=1 in the LPUART_CR1 register.  
TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty  
RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFREQ in the LPUART_RQR register. The RXFNE flag can also be cleared by writing 1 to the RXFREQ in the LPUART_RQR register.  
An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register.  
0: Data is not received  
1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected  
This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE=1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.  
0: No Idle line is detected  
1: Idle line is detected  
*Note*: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).  
*If mute mode is enabled (MME=1), IDLE is set if the LPUART is not mute (RWU=0), whatever the mute mode selected by the WAKE bit. If RWU=1, IDLE is not set.*
54.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20
Reset value: 0x0000 0000

Bit 3 ORE: Overrun error
This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE=1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF in the LPUART_ICR register.
An interrupt is generated if RXNEIE=1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.
1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.
This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 NE: Start bit noise detection flag
This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.
0: No noise is detected
1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.
In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 FE: Framing error
This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 PE: Parity error
This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.
Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag
   Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

*Note:* If the LPUART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag
   Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
   Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag
   Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
   Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
   Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
   Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
   Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
   Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

### 54.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDR[8:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see Section 54.4.1: LPUART block diagram).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

### 54.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28
Reset value: 0x0000 0000

```
31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16
31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16
   TDR[8:0]
```

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value
Contains the data character to be transmitted.
The TDR register provides the parallel interface between the internal bus and the output shift register (see Section 54.4.1: LPUART block diagram).
When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

*Note: This register must be written only when TXE/TXFNF=1.*

### 54.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE=0).
Address offset: 0x2C
Reset value: 0x0000 0000

```
31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16
31  30  29  28  27  26  25  24  23  22  21  20  19  18  17  16
   PRESCALER[3:0]
```

Bits 31:9 Reserved, must be kept at reset value.
Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

- 0000: input clock not divided
- 0001: input clock divided by 2
- 0010: input clock divided by 4
- 0011: input clock divided by 6
- 0100: input clock divided by 8
- 0101: input clock divided by 10
- 0110: input clock divided by 12
- 0111: input clock divided by 16
- 1000: input clock divided by 32
- 1001: input clock divided by 64
- 1010: input clock divided by 128
- 1011: input clock divided by 256

Others: Reserved, must not be used.

**Note:** When **PRESCALER** is programmed with a value different from the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this field is reserved and must be kept at reset value. Refer to Section 54.3: LPUART implementation on page 2500.

54.7.14 LPUART register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 31:4</th>
<th>Bits 30-27</th>
<th>Bits 26-23</th>
<th>Bits 22-19</th>
<th>Bits 18-15</th>
<th>Bits 14-11</th>
<th>Bits 10-7</th>
<th>Bits 6-3</th>
<th>Bits 2-0</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>LPUART_CR1</td>
<td>FIFOEN</td>
<td>RXFIE</td>
<td>TXFIE</td>
<td>M1</td>
<td>M0</td>
<td>DEAT[4:0]</td>
<td>DEDT[4:0]</td>
<td>WAKE</td>
<td>PCE</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RXFFIE</td>
<td>TXFFIE</td>
<td>WUX[1:0]</td>
<td>WUX[1:0]</td>
<td>WUX[1:0]</td>
<td>REV[1:0]</td>
<td>STOP[1:0]</td>
<td>CTSE</td>
<td>DMAR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x04</td>
<td>LPUART_CR2</td>
<td>ADD[7:0]</td>
<td>TCK[1:0]</td>
<td>RXFIE</td>
<td>TXFIE</td>
<td>WUX[1:0]</td>
<td>WUX[1:0]</td>
<td>WUX[1:0]</td>
<td>CTSE</td>
<td>DMAR</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x08</td>
<td>LPUART_CR3</td>
<td>WIREF</td>
<td>WIS[1:0]</td>
<td>WIS[1:0]</td>
<td>OVR[0:1]</td>
<td>OVR[0:1]</td>
<td>OVR[0:1]</td>
<td>CTSE</td>
<td>MTIE</td>
<td>MIDE</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>LPUART_BRR</td>
<td>BRR[19:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Table 574. LPUART register map and reset values

**Reset value**

0x00 LPUART_CR1
FIFO mode enabled

0x04 LPUART_CR2
ADD[7:0]

0x08 LPUART_CR3
FIFO mode enabled
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x10-0x14</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0x18</td>
<td>LPUART_RQR</td>
<td>Reset value</td>
</tr>
<tr>
<td>0x1C</td>
<td>LPUART_ISR FIFO mode enabled</td>
<td>RXFF TXFF RXACK TXACK WFF WIF TXE RXE CTS CMR BUSY</td>
</tr>
<tr>
<td>0x1C</td>
<td>LPUART_ISR FIFO mode disabled</td>
<td>RXFF TXFF RXACK TXACK WFF WIF TXE RXE CTS CMR BUSY</td>
</tr>
<tr>
<td>0x20</td>
<td>LPUART_ICR</td>
<td>WFF CTS CMR BUSY</td>
</tr>
<tr>
<td>0x24</td>
<td>LPUART_RDR</td>
<td>RDR[8:0]</td>
</tr>
<tr>
<td>0x28</td>
<td>LPUART_TDR</td>
<td>TDR[8:0]</td>
</tr>
<tr>
<td>0x2C</td>
<td>LPUART_PRESC</td>
<td>PRESCALE R[3:0]</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
55 Serial peripheral interface (SPI)

55.1 Introduction
The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and is capable of operating in multi slave or multi master configurations. The device configured as master provides communication clock (SCK) to the slave device. The slave select (SS) and ready (RDY) signals can be applied optionally just to setup communication with concrete slave and to assure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

55.2 SPI main features
- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multi slave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Hardware or software management of SS for both master and slave
- Adjustable minimum delays between data and between SS and data flow
- Configurable SS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transaction to control SS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI formats support
- Hardware CRC feature can verify integrity of the communication at the end of transaction by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instance)
- Configurable FIFO thresholds (data packing)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing the slave device ready to handle the data flow
55.3 SPI implementation

The table below describes the SPI implementation. The instances are applied either with a full set or a limited set of features.

<table>
<thead>
<tr>
<th>SPI feature</th>
<th>SPI2S1, SPI2S2, SPI2S3 and SPI2S6 (full feature set instances)</th>
<th>SPI4 and SPI5 (full feature set instances)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data and CRC size</td>
<td>Configurable from 4 to 32 bits</td>
<td>Configurable from 4 to 16 bits</td>
</tr>
<tr>
<td>CRC computation</td>
<td>CRC polynomial length configurable from 5 to 33 bits</td>
<td>CRC polynomial length configurable from 5 to 17 bits</td>
</tr>
<tr>
<td>Size of FIFOs</td>
<td>16x8 bits</td>
<td>8x8 bits</td>
</tr>
<tr>
<td>Number of data control (TSIZE)</td>
<td>Up to 65536</td>
<td>Up to 65536</td>
</tr>
<tr>
<td>I2S feature</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Autonomous in Stop modes with wake-up capability</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Autonomous in LP-Stop and Standby modes with wake-up capability</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Note: For detailed information about instances capabilities to exit from Stop and Standby modes refer to Table 579: SPI wake-up and interrupt requests.
55.4 SPI functional description

55.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in Figure 774.

Figure 774. SPI/I2S block diagram

The simplified scheme of Figure 774 shows three fully independent clock domains:

- the spi_pclk clock domain
- the spi_ker_ck kernel clock domain
- the serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.
The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be setup correctly (for example when underrun or overrun is evaluated, see Section 55.5.2 for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see Section 55.5.1).

When the SPI works as master, it needs **spi_ker_ck** kernel clock coming from RCC active during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed optionally. The signal is then provided to slaves via the SCK pin and internally to the serial interface domain of the master.

### 55.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO**: master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI**: master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK**: serial clock output pin for SPI masters and input pin for SPI slaves.
- **SS**: slave select pin. Depending on the SPI and SS settings, this pin can be used to either:
  - select an individual slave device for communication
  - synchronize the data frame, or
  - detect a conflict between multiple masters
  See Section 55.4.7 for details.
- **RDY**: optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transaction without any risk of data flow corruption. It can be checked by master to control temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features additional I2S specific MCK signal. For more details about I2S signals see Section 55.9.2.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to Table 576 and Table 577 for the list of SPI input / output pins and internal signals.

**Table 576. SPI/I2S input/output pins**

<table>
<thead>
<tr>
<th>Pin name</th>
<th>I/O type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISO/SDI</td>
<td>Input/output</td>
<td>Master data input / slave data output</td>
</tr>
<tr>
<td>MOSI/SDO</td>
<td>Input/output</td>
<td>Master data output / slave data input</td>
</tr>
<tr>
<td>SCK/CK</td>
<td>Input/output</td>
<td>Master clock output / slave clock input</td>
</tr>
</tbody>
</table>
The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software SS management) or three or four wires (with hardware SS management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slave(s) for communication by SS line when it is managed by hardware. The data between the master and the slave flow on the MOSI and/or MISO lines.

### Communications between one master and one slave

The communication flow may use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode or the simplex (two wires) mode. The SS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the SS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to assure correct management of all the transacted data at slave side.
Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 775. Full-duplex single master/ single slave application

1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see Section 55.4.7 for details).
2. RDY signal provided by slave can be read by master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.
1. To apply SS pins interconnection is not mandatory to make the SPI interface working (see Section 55.4.7 for details).

2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs.

3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.

4. RDY signal provided by slave can be read by master optionally.

**Simplex communications**

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pins pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**
  
  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the SS pin (or software managed internal signal) is active (see Section 55.4.7).

- **Receive-only mode: COMM[1:0] = 10**
  
  In master mode, the MOSI output is disabled and may be used as GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the Rx FIFO is full, when the MASRX bit in the SPI_CR1 is set.

  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see Section 55.4.7).

**Note:** In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (This bit may only be modified when the SPI is disabled). Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transaction direction (bidirectional mode is enabled, while the HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.
1. SS pins interconnection is not mandatory to make the SPI interface working (see Section 55.4.7).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. RDY signal provided by slave can be read by master optionally.

55.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see Figure 778).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave SS input (only one slave can control data on common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.
1. Master single SS pin hardware output functionality cannot support this topology (to be replaced by set of GPIOs under SW control) and user should avoid SPI AF setting at the pin (see Section 55.4.7 for details).

2. If the application cannot assure that no more than a single SS active signal is provided by the master at a given time, it is better to configure MISO pins into open-drain configuration with an external pull-up on the MISO line to prevent conflicts between interconnected outputs of the slaves. Else, a push-pull configuration can be applied without extra resistor (see I/O alternate function input/output (GPIO) section).

3. RDY signals can be read by master from the slaves optionally.

The master can handle the SPI communication with all the slaves in time when a circular topology is applied (see Figure 779). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (SS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transacted by the master is equal to the number of slaves.

Then when a first data frame is transacted in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time. Correspondingly, the lastly transacted data finishing the session is dedicated for the first slave node while its firstly outgoing data just reaches the master input, after its circling
around the chain passing through all the other slaves during the session. The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transacted between the receiver and the transmitter by hardware.

In this case, the transmission underrun feature is configured in a mode repeating lastly received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case the underrun happens in fact after this first data frame is transacted. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

**Figure 779. Master and three slaves connected in circular (daisy chain) topology**

1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. RDY signals can be read by master optionally either separately or configured as open drain outputs (while RDIOP=0) and connected together with a pull up resistor as a common chain ready status overdriven by the slowest device.
55.4.6 **Multimaster communication**

Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the SS pin is used configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is completed, the active slave select signal is released and the node mastering the bus temporary returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

**Figure 780. Multimaster application**

1. The SS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control as passive node is configured as a slave.
2. The RDY signal is not used in this communication.

55.4.7 **Slave select (SS) pin management**

In slave mode, the SS works as a standard 'chip select' input and lets the slave communicate with the master. In master mode, the SS can be used either as an output or an input. As an input it can prevent a multi master bus collision, and as an output it can drive a slave select signal of a single slave. The SS signal can be managed internally (software management of the SS input) or externally when both the SS input and output are associated with the SS pin (hardware SS management). The user can configure which level of this input/output external signal (present on the SS pin) is considered as active one by the SSIOP bit setting. SS level is considered as active if it is equal to SSIOP.
The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software SS management (SSM = 1):** in this configuration, slave select information is driven internally by the SSI bit value in the register SPI_CR1. The external SS pin is free for other application uses (as GPIO or other alternate function).

- **Hardware SS management (SSM = 0):** in this case, there are two possible configurations. The configuration used depends on the SS output configuration (SSOE bit in register SPI_CFG2).
  - **SS output enable (SSOE = 1):** this configuration is only used when the MCU is set as master. The SS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply proper TSIZE>0 setting to control the SS output correctly. Even if SPI AF is not applied at the SS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to assure default SS input level and prevent any mode fault evaluation at input of the master SS internal logic applicable at a multimaster topology exclusively.
    a) When SSOM = 0 and SP = 000, the SS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.
    b) When SP = 001, a pulse is generated as defined by the TI mode.
    c) When SSOM = 1, SP = 000 and MIDI > 1 the SS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (1 to 14).
    d) SS input is forced to non active state internally at master to prevent its any mode fault.
  - **SS output disable (SSM = 0, SSOE = 0):**
    a) if the micro-controller is acting as the master on the bus, this configuration allows multi master capability. If the SS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).
    b) In slave mode, the SS pin works as a standard ‘chip select’ input and the slave is selected while the SS line is at its active level.

*Note:* The purpose of automatic switching into slave mode at mode fault condition is to avoid the possible conflicts on data and clock line. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

When the SPI slave is enabled in the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (transition from non-active to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external SS pin is fixed. There is no such protection in the SS software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.
When the hardware output SS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the SS signal between data frames and can insert an extra delay at the beginning of every transaction (to separate the SS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see Figure 782).

Figure 782. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)

2. CPHA = 0, CPOL = 0, SSIOp = 0, LSBFIRST = 0.

Additionally, bit SSOM = 1 setting invokes specific mode which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in Figure 783.
Figure 783. SS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)

I. CPHA=0, CPOL=0, SSIOPL=0, LSBFRST=0

II. CPHA=1, CPOL=0, SSIOPL=0, LSBFRST=0

III. CPHA=0, CPOL=1, SSIOPL=1, LSBFRST=1

IV. CPHA=1, CPOL=1, SSIOPL=1, LSBFRST=1
2. SS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by MIDI parameter). If MIDI is set the frames are separated by single SCK period but no interleaving pulse appears on SS.

55.4.8 Ready pin (RDY) management

The status of the slave capability to handle data, can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transaction. The reason can be that slave's TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, the transaction fails great probably.

The logic to control the RDY output is rather complex, tied closely with TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transaction. This pessimistic logic is suppressed at the end of a transaction only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform master in time that it is necessary to suspend the transaction temporarily till the next transacted data can be processed safely again. When RDY signal input is enabled at master side, master suspends the communication once the slave indicates not ready status. This prevents the master to complete transaction of an ongoing frame which just empties slave's TxFIFO or full fills its Rx FIFO till a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if TSIZE = 0 configuration is applied at slave because slave then never evaluates end of transaction (which suppresses the not ready status just when the last data is sent). Then the user has to release the Rx FIFO and/or write additional (even dummy) data to Tx FIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8-bit.

55.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships may be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data are being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.
If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transacted (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in Figure 784).

Figure 784, shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit. The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 784. Data clock timing diagram

1. The order of data bits depends on LSBFRST bit setting.
Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit in SPI_CFG2 register.

At instance with full feature set, the data frame size is chosen by using the DSIZE[4:0] bits at SPI_CFG1 register. It can be set from 4-bit up to 32-bit length and the setting applies for both transmission and reception. When the SPI_TXDR/SPI_RXDR registers are accessed, data frames are always right-aligned into either a byte (if the data fit into a byte), a half-word or a word (see Figure 785).

If the access is a multiple of the configured data size, data packing is applied automatically. During communication, only bits within the data frame are clocked and transferred.

Figure 785. Data alignment when data size is not equal to 8, 16 or 32 bits

<table>
<thead>
<tr>
<th>DSIZE &lt;= 8-bits data is right aligned on byte Example: DSIZE[4:0]=00011</th>
<th>9-bits &lt;= DSIZE &lt;= 16-bits data is right aligned on half-word Example: DSIZE[4:0]=01101</th>
<th>17-bits &lt;= DSIZE &lt;= 32-bits data is right aligned on word Example: DSIZE[4:0]=11010</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 4 3 0</td>
<td>15 14 13 0</td>
<td>31 27 26 0</td>
</tr>
<tr>
<td>Tx</td>
<td>X X X X</td>
<td>X X</td>
</tr>
<tr>
<td>Rx</td>
<td>0 0 0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>

Note: The minimum data length is 4 bits. If a data length of less than 4 bits is selected, it is forced to an 4-bit data frame size.

At the instance with limited set of features, data size is fixed to multiply 8-bit up to maximum data length (depends on instance) in according to DSIZE[4:3] bits value. If the SPI_TXDR or SPI_RXDR are accessed by wider access (a multiply of the configured the data size), data packing is applied automatically.

55.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, SS and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up proper values of all ‘not reserved’ bits and bitfields, prior SPI is enabled, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MID[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When slave is configured at TI mode, MBR[2:0] setting is considered, too.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] is required if CRCEN is set.
d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI and MIDI are not required in TI mode.
e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless the peripheral enable. So any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to SPI_CRCPOLY and into TCRCINI, RCRCINI and CRC33_17 bits at the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see chapter Communication using DMA).
6. Configure SSI, HDDIR and MASRX at SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CFG1 register if the configuration protection is required (for safety).

55.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while a traffic is ongoing on the bus, assuming that the SS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the SS signal is applied (see Section 55.4.7). The data register of the slave transmitter should contain data to be sent before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by SS, else the following transaction may be desynchronized.

When the SPI slave is enabled at the hardware SS management mode, all the traffics are ignored even in case of the SS is found at active level. They are ignored until the slave detects a start of the SS signal (its transition from non-active to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when external SS pin is fixed. There is no such protection at the SS software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see Section 55.4.15.

55.4.12 SPI data transmission and reception procedures

The setting of data communication format follows the basic principle that sure number of data with a flexible size must be transferred within a session (transaction) while, optionally, the data handling can be cumulated effectively into a single access of the SPI data registers (data packing) or even grouped into a sequence of such services if data is collected at consistent bigger data packets. The data handling services are based upon FIFO packet occupancy events. That is why the complete data packet must be serviced exclusively upon a dedicated packet flag.
To understand better the next detailed content of this section, the user should capture the configuration impact and meaning of the following items at first:

**Data size (DSIZE)** - defines data frame (sets the number of bits at single data frame).

**FIFO threshold (FTHLV)** - defines data packet, sets the number of data frames at single data packet and so the occurrence of the packet occupancy events to handle SPI data registers either by software or by DMA.

**Data access** – a way how to handle the SPI data register content when the transfer data between the application and the SPI FIFOs upon a packet event. It depends on the packet size configuration. Optionally, multiply data can be handled effectively by a single access of the register (by data packing) or by sequence of such accesses (when servicing a bigger data packet).

**FIFO size** – capacity or space to absorb available data. It depends on the data size and the internal hardware efficiency how the data is compressed and organized within this space. The FTHLV setting must respect the FIFO capacity to store two data packets at least.

**Transaction size (TSIZE)** – defines total number of data frames involved at a transaction session overall possibly covered by several data packet services. There is no need to align this number with the packet size (handling of a last not aligned data packet is supported if TSIZE is programmed properly).

**Data handling via RxFIFO and TxFIFO**

All SPI data transactions pass through the embedded FIFOs organized by bytes (N x 8-bit). The size of the FIFOs (N) is dependent on the product and the peripheral instance. This enables the SPI to work in a continuous flow, and prevents overruns when the data frame size is short or the interrupt/DMA latency is too long. Each direction has its own FIFO called TxFIFO and RxFIFO, respectively.

The handling of the FIFOs content is based on servicing data packet events exclusively raised by dedicated FIFO packet occupancy flags (TXP, RXP or DXP). The flags occurrence depends on the data exchange mode (duplex, simplex), the data frame size (number of bits in the frame) and how data are organized at data packets. The frequency of the packet events can be decreased significantly when data are organized into packets via defining the FIFOs threshold. Several data frames grouped at packet can be then handled effectively based on a single FIFO occupancy packet event either by a single SPI data register access or their sequence what consumes less system performance. The user can control the access type by casting the data register address to force a concrete CPU instruction applied for the register read or write. The access then can be 8-bit, 16-bit or 32-bit but single data frame must be always accessed at least. It is crucial to keep the setting of the packet size (FTHLV) and the data size (DSIZE) always balanced with the applied data registers access (no matter if a single access or their sequence is applied) just to apply and complete service of a single data packet upon its event. This principle, occurrence and clearing capabilities of the FIFO occupancy flags are common no matter if DMA, interrupt, or polling is applied.

A read access to the SPI_RXDR register returns the oldest value stored in the RxFIFO that has not been read yet. A write access to the SPI_TXDR stores the written data in the TxFIFO at the end of a send queue.

A read access to the SPI_RXDR register must be managed by the RXP event. This flag is set by hardware when at least one complete data packet (defined as receiver threshold by FTHLV[3:0] bits at the SPI_CFG1 register) is available at the reception FIFO while reception is active. The RXP is cleared as soon as less data than complete single packet is available in the RxFIFO, when reading SPI_RXDR by software or by DMA.
The RXP triggers an interrupt if the RXPIE bit is set and/or a DMA request if the RXDMAEN bit is set.

Upon setting of the RXP flag, the application performs the due number of SPI data register reads to download the content of one data packet. Once a complete data packet is downloaded, the application software or DMA checks the RXP value to see if other packets are pending into the receive FIFO and, if so, downloads them packet by packet until the RXP reads 0. Rx_FIFO can store up to N data frames (for frame size ≤ 8-bit), N/2 data frames (for 8-bit < frame ≤ 16-bit), N/3 data frames (for 16-bit < frame ≤ 24-bit) or N/4 data frames (if data frame > 24-bit) where N is the size of the FIFO in bytes.

At the end of a reception, it may happen that some data are still available in the Rx_FIFO, without reaching the FTHLV level, thus the RXP is not set. In this case, the number of remaining RX data frames in the FIFO is indicated by RXWNE and RXPLVL fields in the SPI_SR register. It happens when the number of the last data received in a transfer cannot fully accomplish the configured packet size; in case the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of reads from the Rx_FIFO used for the previous complete data packets without drawbacks: only the consistent data (completed data frames) are popped from the Rx_FIFO while redundant reads (or any uncompleted data) are reading 0. Thanks to that, the application software can treat all the data in a transfer in the same way, and is off-loaded to foresee the reception of the last data in a transfer and to calculate the due number of reads to be popped from Rx_FIFO.

In a similar way, the write access of a data frame to be transmitted is managed by the TXP event. This flag is set by hardware when there is enough space for the application to push at least one complete data packet (defined at FTHLV[3:0] bits at SPI_CFG1 register) into the transmission FIFO while transmission is active. The TXP is cleared as soon as the Tx_FIFO is filled by software and/or by the DMA. The space currently available for any next complete data packet is lost. This can lead to oscillations of the TXP signal when data are released out from the Tx_FIFO while a new packet is stored frame by frame. Any write to the Tx_FIFO is ignored when there is no sufficient room to store at least a single data frame (TXP event is not respected), when TXTF is set or when the SPI is disabled.

The TXP triggers an interrupt if the TXPIE bit is set and/or with a DMA request if the TXDMAEN bit is set. The TXPIE mask is cleared by hardware when the TXTF flag is set.

Upon setting of the TXP flag, the application performs the due number of SPI data register writes to upload the content of one entire data packet. Once new complete data packet is uploaded, the application software or DMA checks the TXP value to see if other packets can be pushed into the Tx_FIFO and, if so, uploads them packet by packet until TXP reads 0.

The number of last data in a transfer can be shorter than the configured packet size in the case when the transfer size and the packet size are not aligned. Nevertheless the application software can still perform the standard number of data register writes used for the previous packets without drawbacks: only the consistent data are pushed into the Tx_FIFO while redundant writes are discarded. Thanks to that, the application software can treat all the data in a transfer in the same way and is off-loaded to foresee the transmission of the last data in a transfer and from calculating the due number of writes to push the last data into Tx_FIFO. Just for the last data case, the TXP event is asserted by SPI once there is enough space into Tx_FIFO to store remaining data to complete current transfer.

Both TXP and RXP events can be polled or handled by interrupts. The DXP bit can be monitored as a common TXP and RXP event at full-duplex mode.
Upon setting of the DXP flag the application performs the due number of writes to the SPI data register to upload the content of one entire data packet for transmission, followed by the same number of reads from the SPI data register to download the content of one data packet. Once one data packet is uploaded and one is downloaded, the application software or DMA checks the DXP value to see if other packets can be pushed and popped in sequence and, if so, uploads/downloads them packet by packet until DXP reads 0.

The DXP triggers an interrupt if the DXPIE bit is set. The DXPIE mask is cleared by hardware when the TXTF flag is set.

The DXP is useful in full-duplex communication in order to optimize performance in data uploading/downloading, and reducing the number of interrupts or DMA sequences required to support an SPI transfer thus minimizing the request for CPU bandwidth and system power especially when SPI is operated in Stop mode.

When relay on the DXP interrupt exclusively, the user must consider the drawback of such a simplification when TXP and RXP events are serviced by common procedures because the TXP services are delayed by purpose in this case. This is due to fact that the TXP events precedes the reception RXP ones normally to allow the TXP servicing prior transaction of the last frame fully emptying the TxFIFO else master cannot provide a continuous SCK clock flow and the slave can even face an underrun condition. The possible solution is to prefill the TxFIFO by few data packets ahead prior the session starts and to handle all the data received after the TXTF event by EOT exclusively at the end of the transaction (as TXTF suppresses the DXP interrupts at the end of the transaction). In case of CRC computation is enabled, the user must calculate with additional space to accommodate the CRC frame at RxFIFO when relying on EOT exclusively at the end of transaction.

Another way to manage the data exchange is to use DMA (see Section 55.4.15).

If the next data is received when the RxFIFO is full, an overrun event occurs (see description of OVR flag in Section 55.5.2). An overrun event can be polled or handled by an interrupt.

This may happen in slave mode or in a master receive mode when MASRX = 0. If MASRX bit is set at a master receiver, the generated clock stops automatically when the RxFIFO is full, therefore overrun is prevented.

Both RxFIFO and TxFIFO content is kept flushed and cannot be accessed when SPI is disabled (SPE = 0).

**Transaction handling**

A few data frames can be passed at single sequence to complete a message. The user can handle a number of data within a message thanks to the value stored into TSIZE. In principle, the transaction of a message starts when the SPI is enabled by setting CSTART bit and finishes when the total number of required data is transacted. The end of transaction controls the CRC and the hardware SS management when applied. To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed.

If TSIZE is kept at zero while CSTART is set, an endless transaction is initialized (no control of transfer size is applied). During an endless transaction, the number of transacted data aligned with FIFOs threshold is supported exclusively. If the number of data (or its grouping into packets) is unpredictable, the user must keep the FIFO threshold setting (packet size) at single data (FTHLV = 0) to assure that each data frame raises its own packet event to be serviced by the application or DMA. The transaction can be suspended at any time thanks
to CSUSP which clears the CSTART bit. SPI must be always disabled after such software suspension and re-enabled before the next transaction starts.

When the transmission is enabled, a sequence begins and continues while any data is present in the TxFIFO of the master. The clock signal is provided permanently by the master until TxFIFO becomes empty, then it stops, waiting for additional data.

In receive-only modes, half-duplex (COMM[1:0] = 11, HDDIR = 0) or simplex (COMM[1:0] = 10) modes, the master starts the sequence when the SPI is enabled and the transaction is released by setting the CSTART bit. The clock signal is provided by the master and it does not stop until either SPI or receive-only mode is disabled/suspended by the master. The master receives data frames permanently up to this moment. The reception can be suspended either by software control, writing 1 to the CSUSP bit in the SPI_CR1 register, or automatically when MASRX = 1 and RxFIFO becomes full or upon the RDY status if this signal is applied (see Section 55.4.8). The reception is automatically stopped also when the number of frames programmed in TSIZE has been completed.

In order to disable the master receive-only mode, the SPI must first be suspended. When the SPI is suspended, the current frame is completed, before changing the configuration.

**Caution:**

If the SPE bit is cleared in master mode, while the reception is ongoing without any suspending, the clock is stopped without completing the current frame, and the RxFIFO is flushed.

While the master can provide all the transactions in continuous mode (SCK signal is continuous) it must respect the slave capability to handle data flow and its content at anytime. If the slave features the RDY signal option, the master can monitor the RDY signal issued by the slave, to control the communication flow. If the RDY pin is not used, the slave is considered always ready for communication with the master.

When necessary, the master must slow down the communication and provide either a slower clock or separate frames or data sessions with sufficient delays by MIDI[3:0] bits setting or provide an initial delay by setting MSS[1:0], which postpones any transaction start to give slave sufficient room for preparing data. Be aware data from the slave are always transacted and processed by the master even if the slave cannot prepare it correctly in time. It is preferable for the slave to use DMA, especially when data frames are short, FIFO is accessed by bytes and the SPI bus rate is high.

In order to add some software control on the SPI communication flow from a slave transmitter node, a specific value written in the SPI_UDRD (SPI Underrun Data Register) may be used. On slave side, when TxFIFO becomes empty, this value is sent out automatically as next data and may be interpreted by software on the master receiver side (either simply dropped or interpreted as a XOFF like command, in order to suspend the master receiver by software).

In the multislave star topology, only a single slave only can be enabled for output data at a given time. The slave just selected for the communication with the master needs to detect a change of its SS input into active level before communication with the master starts. In a single slave system it is not necessary to control the slave with SS, but it is often better to provide the pulse here too, to synchronize the slave with the beginning of each data sequence. The SS can be managed by both software and hardware (Section 55.4.7).
55.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transactions may be corrupted.

In slave mode, the SPI communication can continue when the `spi_pclk` and `spi_ker_ck` clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The `spi_pclk` can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transaction when it stops providing data for transmission. In this case, the clock stops after the last data transaction. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, in order to stop the peripheral, the SPI communication must first be suspended, by setting the `CSUSP` bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must be always disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the `RXP`, `RXWNE` and `RXPLVL` fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transactions, for example:

- When the master handles SS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide proper end of SS pulse for slave, or
- When transaction streams from DMA or FIFO are completed while the last data frame or CRC frame transaction is still ongoing in the peripheral bus.

When `TSIZE>`0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check end of the bus activity. When `TSIZE = 0`, the user has to check TXC, SUSP or FIFO occupancy flags in according with the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed. TXC/EOT is set when CRC frame is completed in this case. When a transmission is suspended the software has to wait till CSTART bit is cleared.
2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).
In slave mode, any on going data are lost when disabling the SPI.

**Controlling the I/Os**

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR setting. When active output control is applied (AFCNTR = 1) and SPI is just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like SS and SCK at master and RDY at slave) can toggle immediately to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). The data line output (MOSI at master and MISO at slave) can instead change its level immediately at dependency on the actual Tx_FIFO content with the effect of potentially making invalid and no more guaranteed the value of the latest transacted bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transaction is completed.

*Note:* Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done for example by inserting small software delay between EOT event occurrence and SPI disable action.

### 55.4.14 Data packing

From user point of view there are two ways of data packing which can overlay each other:

- **Type of access when data are written to TxFIFO or read from Rx FIFO**
  
  Multiple data can be pushed or fetched effectively by single access if data size is multiplied less than the access performed upon SPI_TXDR or SPI_RXDR registers.

- **Number of data to be handled during the single software service**
  
  It is convenient to group data into packets and cumulate the FIFO services overall the data packet content exclusively instead of handling data frame by frame separately. The user can define packets by FIFO threshold settings. Then all the FIFO occupancy events are related to that threshold level while required services are signalized by proper flags with interrupt and/or wake-up capabilities.

When the data frame size fits into one byte (less than or equal to 8 bits), the data packing is used automatically when any read or write 16-bit or 32-bit access is performed on the SPI_RXDR/SPI_TXDR register. The multiple data frame pattern is handled in parallel in this case. At first, the SPI operates using the pattern stored in the LSB of the accessed word, then with the other data stored in the MSB.

*Figure 786* provides an example of data packing mode sequence handling at full feature set instance. While DSIZE[4:0] is configured to 4-bit there, two or four data frames are written to the Tx_FIFO after the single 16-bit or 32-bit access to the SPI_TXDR register of the transmitter. When the data frame size is between 9-bit and 16-bit, data packing is used automatically when a 32-bit access is done. Least significant half-word is used first (regardless of the LSBFRST value).

This sequence can generate two or four RXP events in the receiver if the Rx_FIFO threshold is set to one frame and data is read on a frame basis, unpacked), or it can generate a single RXP event if the FTHLV[3:0] field in the SPI_CFG1 register is programmed to a multiple of the frames to be read in a packed mode (16-bit or 32-bit read access).

The data are aligned in accordance with *Figure 785*. The valid bits are performed on the bus exclusively. Unused bits are not cared at transmitter while padded by zeros at receiver.
When short data frames (< 8-bit or < 16-bit) are used together with a larger data access mode (16-bit or 32-bit), the FTHLV value must be programmed as a multiple of the number of frames/data access (multiple of 4 if 32-bit access is used to up to 8-bit frames or multiple of 2 if 16-bit access is used to up to 8-bit frames or 32-bit access to up to 16-bit frames.).

The RxFIFO threshold setting must be always higher or equal at least than the following read access size, as spurious extra data would be read otherwise.

The FIFO data access less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

A specific problem appears if an incomplete data packet is available at FIFO: less than threshold set at FTHLV bits.

There are two ways of dealing with this problem:

A. without using TSIZE field

On transmitter side, writing the last data frame of any odd sequence with an 8-bit/16-bit access to SPI_TXDR is enough.

On receiver side, the remaining data may be read by any access. Any extra data read are padded with zeros. Polling the RXWNE and RXPLVL may be used to detect when the RX data are available in the RxFIFO. (A time out may be used at system level in order to detect the polling)

B. using the TSIZE field

On transmitter side, the transaction is stopped by the master when it faces EOT event.

In reception, the RXP flag is not set when EOT is set. In the case when the number of data to be received (TSIZE) is not a multiple of packet size, the number of remaining data is indicated by the RXWNE and RXPLVL fields in the SPI_SR register. The remaining data can be read by any access. Any extra read is padded by zeros.

Figure 786. Packing data in FIFO for transmission and reception at full feature set instance

1. DSIZE[4:0] is configured to 4-bit, data is right aligned, valid bits are performed only on the bus, their order depends on LSBFRST. If it is set, the order is reversed at all the data frames.
55.4.15  Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits in the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs series of reads from the SPI_RXDR register. When EOT is set at the end of transaction and last data packet is incomplete then DMA request is activated automatically in according with RXWNE and RXPLVL[1:0] setting to read rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.
If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.
If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set at SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the spi_pclk in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:
  1. Enable DMA Rx buffer in the RXDMAEN bit in the SPI_CFG1 register, if DMA Rx is used.
  2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
  3. Enable DMA Tx buffer in the TXDMAEN bit in the SPI_CFG1 register, if DMA Tx is used.
  4. Enable the SPI by setting the SPE bit.

To close communication it is mandatory to follow these steps in order:
  1. Disable DMA request for Tx and Rx in the DMA registers, if the DMA issued.
  2. Disable the SPI by following the SPI disable procedure.
  3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits in the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.
Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

If the DMA channel PSIZE value is equal to 16-bit and the SPI data size is less than or equal to 8-bit, then the packing mode is enabled. Similarly, if the DMA channel PSIZE value is equal to 32-bit and the SPI data size is less than or equal to 16-bit, then the packing mode is enabled. The DMA then automatically manages the write operations to the SPI_TXDR register.

Regardless data packing mode is used and the number of data to transfer is not a multiple of the DMA data size (16-bit or 32-bit) while the frame size is smaller, DMA completes the transfer automatically in accordance with the TSIZE field setting.

Alternatively, last data frames may be written by software, in the single/unpacked mode.

To configure any DMA data access less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

55.5 SPI specific modes and control

55.5.1 TI mode

With a specific SP[2:0] bit field setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and SS signals polarity, phase and flow as well as the bits order are fixed so the setting of CPOL, CPHA, LSBFRST, SSOM, SSQE, SSIOP, SSM, RDIOP, RDIOM, MSSI and MIDI is not required when the SPI is in TI mode configuration. The SS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in Figure 787.

In slave mode, the clock generator is used to define time when the slave output at MISO pin becomes to HiZ when the current transaction finishes. The master baud rate setting (MBR[2:0] at SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become HiZ (TRELEASE) depends on internal re-synchronization, too, which takes next additional 2-4 periods of the clock signal feeding the generator. It is given by formula:

$$T_{RELEASE} = \frac{DSIZE[4:0] + 1}{\text{clock frequency}}$$
If the slave detects misplaced SS pulse during data transaction the TIFRE flag is set.

55.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and interrupt is enabled by setting the corresponding Interrupt Enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all data transmitted subsequently are lost. OVR flag triggers an interrupt if OVRIE bit is set.

Clearing the OVR bit is done by a writing 1 to the OVRC bit in the SPI_IFCR. To prevent any next overrun event the clearing should be done after RxFIFO is emptied by software reads. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depends on the UDRCFG bit. When an underrun is detected by slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between last valid data and constant pattern defined at the UDRDR register (see Figure 788). The second configuration (UDRCFG=1) can be used at circular topography structure (see Figure 779). Assuming that TxFIFO is not empty when master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the lastly received data immediately.

The standard transmission is re-enabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. The user should write some data into TxFIFO prior clearing the UDR flag to prevent any next underrun condition occurrence capture.

The data transacted by slave is unpredictable especially when the transaction starts or continues while TxFIFO is empty and underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transaction with a
defined size just starts. First bits can be corrupted in this case, as well, when slave software writes first data into the empty TxFIFO too close prior the data transaction starts (propagation of the data into TxFIFO takes few APB clock cycles).

**Figure 788. Optional configurations of the slave behavior when an underrun condition is detected**

**Mode fault (MODF)**

Mode fault occurs when the master device has its internal SS signal (SS pin in SS hardware mode, or SSI bit in SS software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero till MODF bit is set. This disables SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.
MODF is cleared by writing 1 to the MODFC bit in the SPI_IFCR.

To avoid any multiple slave conflicts in a system comprising several MCUs, the SS pin must be pulled to its non-active level before re-enabling the SPI, by setting the SPE bit.

As a security, hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device the MODF bit cannot be set except as the result of a previous multi master conflict.

A correct software procedure when master overtakes the bus at multi master system should be the following one:

- Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode)
- Put GPIO pin dedicated for another master SS control into active level
- Perform data transaction
- Put GPIO pin dedicated for another master SS control into non active level
- Switch back to slave mode

**CRC error (CRCE)**

This flag is used to verify the validity of the value received when the CRCEN bit in the SPI_CFG1 register is set. The CRCE flag in the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit in the SPI_IFCR.

**TI mode frame format error (TIFRE)**

A TI mode frame format error is detected when an SS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the SS pulse is ignored, and the SPI waits for the next SS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit in the SPI_IFCR. If the TIFREIE bit is set, an interrupt is generated on the SS error detection. As data consistency is no longer guaranteed, communication should be re-initiated by software between master and slave.

### 55.5.3 CRC computation

Two separate 33-bit or two separate 17-bit CRC calculators are implemented to check the reliability of transmitted and received data. For instances with full feature set, the SPI offers CRC polynomial length from 5 to 33 bits when maximum data size is 32-bit and from 9 to 17 bits for the peripheral instances with data size limited to 16 bits. For instances with limited set of features, the CRC polynomial length can be set either to 9 or 17 only when data size is limited to 16 bit and optionally to 33 when data size is extended to 32-bit.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string. For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 +
x^12 + x^5 + 1, write 0x11021 to the SPI_CRCPOLY register for a 32-bit instance, whereas to obtain the full size for a 16-bit instance, write 0x1021 with the CRC33_17 bit set.

The CRCSIZE field in the SPI_CFG1 then defines how many the most significant bits from CRC calculation registers are transacted and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size while its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC pattern size, else the bits unused at the calculation are transacted and expected all zero at the end of the CRC pattern if its size is set greater then the polynomial length.

**CRC principle**

The CRC calculation is enabled by setting the CRCEN bit in the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits in the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

**CRC transfer management**

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transacted, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

No matter what is the CRCSIZE configuration, TSIZE cannot be set neither to 0xFFFF at full feature set instance nor to 0x3FF value at limited feature one if CRC is enabled.

In transmission, the CRC computation is frozen during CRC transaction and the TXCRC are transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when desired number of data is transacted. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is completed, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag in the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.
Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode in order to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI at SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

55.6 SPI in low-power modes

Table 578. Effect of low-power modes on the SPI

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect. SPI interrupts cause the device to exit Sleep mode.</td>
</tr>
<tr>
<td>Stop(1)</td>
<td>The SPI registers content is kept.</td>
</tr>
<tr>
<td>Standby</td>
<td>The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.</td>
</tr>
</tbody>
</table>

1. Refer to Section 55.3: SPI implementation for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

55.7 SPI interrupts

Table 579 gives an overview of the SPI events capable to generate interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.
### Table 579. SPI wake-up and interrupt requests

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable Control bit</th>
<th>Event clear method</th>
<th>Exit from Stop and Standby modes capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)</td>
<td>TXP</td>
<td>TXPIE</td>
<td>TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Data received in RxFIFO (one data packet available - FIFO threshold)</td>
<td>RXP</td>
<td>RXPIE</td>
<td>RXP cleared by hardware when RxFIFO contains less than FTHLV samples</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Both TXP and RXP active</td>
<td>DXP</td>
<td>DXPIE</td>
<td>When TXP or RXP are cleared</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Transmission Transfer Filled</td>
<td>TXTF</td>
<td>TXTFIE</td>
<td>Writing TXTFC to 1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Underrun</td>
<td>UDR</td>
<td>UDRIE</td>
<td>Writing UDRC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Overrun</td>
<td>OVR</td>
<td>OVRIE</td>
<td>Writing OVR to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>CRC Error</td>
<td>CRCE</td>
<td>CRCEIE</td>
<td>Writing CRCEC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>TI Frame Format Error</td>
<td>TIFRE</td>
<td>TIFREIE</td>
<td>Writing TIFREC to 1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Mode Fault</td>
<td>MODF</td>
<td>MODFIE</td>
<td>Writing MODFC to 1</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>End Of Transfer (full transfer sequence completed - based on TSIZE value)</td>
<td>EOT</td>
<td>EOTIE</td>
<td>Writing EOTC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Master mode suspended</td>
<td>SUSP</td>
<td></td>
<td>Writing SUSPC to 1</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>TxFIFO transmission complete (TxFIFO empty)</td>
<td>TXC(3)</td>
<td></td>
<td>TXC cleared by hardware when a transmission activity starts on the bus</td>
<td>No</td>
</tr>
</tbody>
</table>

1. All the interrupt events are capable to wake up system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to ‘functionalities depending on the working mode’ table.

2. Refer to Section 55.3: SPI implementation for information about Standby mode availability.

3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.
55.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length may be 16, 24 or 32 bits\(^{(a)}\)
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, Overrun and Frame Error
- Embedded Rx and TxFIFOs
- Supported I2S protocols:
  - I2S Philips standard
  - MSB-Justified standard (Left-Justified)
  - LSB-Justified standard (Right-Justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - \(F_{MCK} = 256 \times F_{WS}\) for all I2S modes
  - \(F_{MCK} = 128 \times F_{WS}\) for all PCM modes

Note: \(F_{MCK}\) is the master clock frequency and \(F_{WS}\) is the audio sampling frequency.

55.9 I2S functional description

55.9.1 I2S general description

The block diagram shown on Figure 774 also applies for I2S mode.

The SPI/I2S block can work on I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA and parts of interrupt signaling are shared with SPI function. The low-power mode function is also available in I2S mode, refer to Section 55.6: SPI in low-power modes and Section 55.10: I2S interrupts.

\(a\). Not always available, refer to Section 55.3: SPI implementation in order to check if 24 and 32-bit data width are supported.
55.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: Serial Data Output (mapped on the MOSI pin) to transmit the audio samples in master, and to receive the audio sample in slave. Refer to Section: Serial data line swapping on page 2600.
- SDI: Serial Data Input (mapped on the MISO pin) to receive the audio samples in master, and to transmit the audio sample in slave. Refer to Section: Serial data line swapping on page 2600.
- WS: Word Select (mapped on the SS pin) is the frame synchronization. It is configured as output in master mode, and as input for slave mode.
- CK: Serial Clock (mapped on the SCK pin) is the serial bit clock. It is configured as output in master mode, and as input for slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:
- MCK: Master Clock (mapped separately) is used when the I2S is configured in master mode.

55.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = ‘1’), some bitfields are no longer relevant, and must be forced to a specific value in order to guarantee the behavior of the I2S/PCM function. Table 580 shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 580. Bitfields usable in PCM/I2S mode

<table>
<thead>
<tr>
<th>Register name</th>
<th>Bitfields usable in PCM/I2S Mode</th>
<th>Constraints on other bitfields</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI/I2S control register 1 (SPI_CR1)</td>
<td>IOLOCK, CSUSP, CSTART, SPE</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S control register 2 (SPI_CR2)</td>
<td></td>
<td>Set to reset value</td>
</tr>
<tr>
<td>SPI/I2S configuration register 1 (SPI_CFG1)</td>
<td>TXDMAEN, RXDMAEN, FTHLV</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S configuration register 2 (SPI_CFG2)</td>
<td>AFCNTR, LSBFRST, IOSWP</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S interrupt enable register (SPI_IER)</td>
<td>TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S status register (SPI_SR)</td>
<td>SUSP, TIFRE, OVR, UDR, TXP, RXP</td>
<td>Other flags not relevant</td>
</tr>
<tr>
<td>SPI/I2S interrupt/status flags clear register (SPI_IFCR)</td>
<td>SUSPC, TIFREC, OVRC, UDRC</td>
<td>Other fields set to their reset values</td>
</tr>
<tr>
<td>SPI/I2S receive data register (SPI_RXDR)</td>
<td>The complete register</td>
<td></td>
</tr>
</tbody>
</table>
The SPI/I2S block supports master and slave mode for both I2S and PCM protocols.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

In order to improve the robustness of the SPI/I2S block in slave mode, the peripheral re-synchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected.

Note: This re-synchronization mechanism is not available for the I2S LSB justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

### Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits in the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel(a)
- 32-bit data packed in a 32-bit channel(a)

#### Table 580. Bitfields usable in PCM/I2S mode (continued)

<table>
<thead>
<tr>
<th>Register name</th>
<th>Bitfields usable in PCM/I2S Mode</th>
<th>Constraints on other bitfields</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI/I2S polynomial register (SPI_CRC POLY)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S transmitter CRC register (SPI_TXCRC)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S receiver CRC register (SPI_RXCRC)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S underrun data register (SPI_UDRDR)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPI/I2S configuration register (SPI_I2SCFGR)</td>
<td>The complete register</td>
<td>-</td>
</tr>
</tbody>
</table>
In PCM master mode, three frames formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

**Figure 789. Waveform examples**

For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to Section 55.3: SPI implementation in order to check if 24 and 32-bit data width are supported.

**I²S Philips standard**

The I²S Philips standard is selected by setting I²SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I²S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

---

*Not always available, refer to Section 55.3: SPI implementation in order to check if 24 and 32-bit data width are supported.*
Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 55.3: SPI implementation for the supported data sizes.

CKPOL is cleared in order to match the I2S Philips protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

Figure 790 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See Control of the WS Inversion for information concerning the handling of WS signal.

Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 55.3: SPI implementation for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.
MSB justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high, and the right channel if WS is low.

**Figure 792. Master MSB Justified 16-bit or 32-bit full-accuracy length**

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 55.3: SPI implementation to check the supported data size.

CKPOL is cleared in order to match the I2S MSB justified protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

See Control of the WS Inversion for information concerning the handling of WS signal.

**Figure 793. Master MSB justified 16 or 24-bit data length**

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 55.3: SPI implementation to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit the remaining bits are forced to the value of the first bit of the next data to be generated in order to avoid timing issues (see Figure 794).
1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 55.3: SPI implementation to check the supported data size.

**LSB justified standard**

This standard is similar to the MSB justified standard in master mode (no difference for the 16 and 32-bit full-accuracy frame formats). The LSB justified 16 or 32-bit full-accuracy format give similar waveforms than MSB justified mode (see Figure 792) because the channel and data have the same length.

*Note:* In the LSB justified format, only 16 and 32-bit channel length are supported in master and slave mode. This is due to the fact that it is not possible to transfer properly the data if the channel length is not known by transmitter and receiver side.

**Figure 795. LSB justified 16 or 24-bit data length**

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 55.3: SPI implementation to check the supported data size.

CKPOL is cleared in order to match the I2S LSB justified protocol. See Selection of the CK sampling edge for information concerning the handling of WS signal.

See Control of the WS Inversion for information concerning the handling of WS signal.
**PCM standard**

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit in SPI_I2SCFGR register.

In PCM long frame:
- The assertion time of WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

In PCM short frame:
- The assertion time of WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:
- The first data bit is MSb or LSb depending on LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to WSINV bit. See Control of the WS Inversion for information concerning the handling of WS signal.

![Figure 796. Master PCM when the frame length is equal the data length](image)

A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

![Figure 797. Master PCM standard waveforms (16 or 24-bit data length)](image)

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 55.3: SPI implementation to check the supported data size.
If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in Figure 798, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

**Figure 798. Slave PCM waveforms**

1. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), Refer to Section 55.3: SPI implementation to check the supported data size.

**Note:** In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits are forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

**Note:** In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

55.9.6 Additional serial interface flexibility

**Variable frame length in slave**

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for I2S LSB justified protocol.
Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into SPI/I2S configuration register **2 (SPI_CFG2)**.

Selection of the CK sampling edge

The CKPOL bit located into **SPI/I2S configuration register (SPI_I2SCFGR)** allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS Inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips Standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips Standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located into **SPI/I2S configuration register (SPI_I2SCFGR)**.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of **SPI/I2S configuration register 2 (SPI_CFG2)**.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (In order to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

**Table 581** shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS depends also on the protocol selected.
Note: The bit AFCNTR must not be set, when the SPI2S is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of SPI/I2S configuration register 2 (SPI_CFG2), and on the slave/master mode. Table 582 gives details on this feature.

<table>
<thead>
<tr>
<th>Direction</th>
<th>IOSWP</th>
<th>Master mode</th>
<th>Slave mode</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Input line</td>
<td>Output line</td>
</tr>
<tr>
<td>RX</td>
<td>0</td>
<td>SDI</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SDO</td>
<td>-</td>
</tr>
<tr>
<td>TX</td>
<td>0</td>
<td>-</td>
<td>SDO</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>-</td>
<td>SDI</td>
</tr>
<tr>
<td>Full-duplex</td>
<td>0</td>
<td>SDI</td>
<td>SDO</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SDO</td>
<td>SDI</td>
</tr>
</tbody>
</table>

55.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write into the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

When the application wants to use the SPI/I2S block the user has to proceed as follow:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to Section 55.9.16 for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC and UDRC bits of SPI_IFCR register. Note that if the flag SUSP is not cleared (via SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit, in order to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO. The generation of MCK can also be started when SPE goes to 1.
5. Set bit CSTART, in order to activate the serial interface.

As shown in Figure 799, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1.
and the TxFIFO is not empty. Note that the bit clock CK is activated 4 rising edges before the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

**Figure 799. Startup sequence, I2S Philips standard, master**

1. As shown in the figure, MCK can be enabled as soon as the bit SPE is set. It is generated after a synchronization delay (SYNC_DLY1). See MCK generation in Section 55.9.9: Clock generator for more information.

2. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

**Note:** Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about 3 periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For I2S Philips standard, the start-of-frame condition is a falling edge of WS signal. The transmission/reception starts one bit clock later. If WSINV = 1, then the start-of-frame condition is a rising edge.

- For other protocols, the start-of-frame condition is a rising edge of WS signal. The transmission/reception starts at rising edge of WS for MSB aligned protocol. The transmission/reception starts one bit clock later for PCM protocol. If WSINV = 1, then the start-of-frame condition is a falling edge.

**Figure 800** shows an example of startup sequence in I2S Philips standard, slave mode.
55.9.8 Stop sequence

The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

The Figure 801 shows an example of stop sequence in the case of master mode. The CSUSP bit is set, during the transmission of left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear SPE bit.

Note: In slave mode, the stop sequence is only controlled by the SPE bit.

Figure 800. Startup sequence, I2S Philips standard, slave

Figure 801. Stop sequence, I2S Philips standard, master

Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after 2 periods of CK clock (SYNC_DLY).
55.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the Frame Synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency. If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

![Figure 802. I2S clock generator architecture](image)

The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

**Note:** *In master mode, there is no specific constraints on the ratio between the bus clock rate (\(F_{bus}\)) and the bit clock (\(F_{bit}\)). The bus clock frequency must be high enough in order to support the data throughput.*

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

\[
F_W = \frac{F_{i2s\_clk}}{256 \times (2 \times I2SDIV + ODD)}
\]

and by this formula in PCM mode:

\[
F_W = \frac{F_{i2s\_clk}}{128 \times (2 \times I2SDIV + ODD)}
\]

In addition, the frequency of the MCK (\(F_{MCK}\)) is given by the formula:

\[
F_M = \frac{F_{i2s\_clk}}{(2 \times I2SDIV + ODD)}
\]

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

\[
F_W = \frac{F_{i2s\_clk}}{32 \times (CHLEN + 1) \times (2 \times I2SDIV + ODD)}
\]
And by this formula in PCM mode:

\[
F_{WS} = \frac{F_{i2s\_clk}}{16 \times (\text{CHLEN} + 1) \times (2 \times \text{I2SDIV} + \text{ODD})}
\]

Where \(F_{WS}\) is the frequency of the frame synchronization, and \(F_{i2s\_clk}\) is the frequency of the kernel clock provided to the SPI/I2S block.

**Note:** CHLEN and ODD can be either 0 or 1. I2SDIV can take any values from 0 to 255 when ODD = 0, but when ODD = 1, the value I2SDIV = 1 is not allowed. When I2SDIV = 0, then \((2 \times \text{I2SDIV}) + \text{ODD}\) is forced to 1.

**Note:** When \((2 \times \text{I2SDIV}) + \text{ODD}\) is odd, the duty cycle of MCK or the CK signals are not 50%. Care must be taken when odd ratio is used: it can impact margin on setup and hold time. For example if \((2 \times \text{I2SDIV}) + \text{ODD}\) = 5, then the duty cycle can be 40%.

Table 583 provides examples of clock generator programming for I2S modes.

### MCK generation

The master clock MCK is generated when the following conditions are met:
- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,
- SPE must be set

<table>
<thead>
<tr>
<th>(i2s_clk) (MHz)</th>
<th>Channel length (bits)</th>
<th>I2SDIV</th>
<th>ODD</th>
<th>MCK</th>
<th>Sampling rate: (F_{WS}) (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.288</td>
<td>16</td>
<td>12</td>
<td>0</td>
<td>No</td>
<td>16</td>
</tr>
<tr>
<td>12.288</td>
<td>32</td>
<td>6</td>
<td>0</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>12.288</td>
<td>16</td>
<td>6</td>
<td>0</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>12.288</td>
<td>32</td>
<td>3</td>
<td>0</td>
<td></td>
<td>32</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>16</td>
<td>0</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>8</td>
<td>0</td>
<td></td>
<td>48</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>8</td>
<td>0</td>
<td></td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>4</td>
<td>0</td>
<td></td>
<td>96</td>
</tr>
<tr>
<td>49.152</td>
<td>16</td>
<td>4</td>
<td>0</td>
<td></td>
<td>192</td>
</tr>
<tr>
<td>49.152</td>
<td>32</td>
<td>2</td>
<td>0</td>
<td></td>
<td>192</td>
</tr>
</tbody>
</table>
55.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 803 shows the allowed settings between APB access sizes, DATFMT and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 803. Data Format

Table 583. CLKGEN programming examples for usual I2S frequencies (continued)

<table>
<thead>
<tr>
<th>i2s_clk (MHz)</th>
<th>Channel length (bits)</th>
<th>I2SDIV</th>
<th>ODD</th>
<th>MCK</th>
<th>Sampling rate: F_WS (kHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.096</td>
<td>16 or 32</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>16</td>
</tr>
<tr>
<td>24.576</td>
<td>16 or 32</td>
<td>3</td>
<td>0</td>
<td>-</td>
<td>32</td>
</tr>
<tr>
<td>49.152</td>
<td>16 or 32</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>48</td>
</tr>
<tr>
<td>12.288</td>
<td>16 or 32</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>32</td>
</tr>
<tr>
<td>61.44</td>
<td>16 or 32</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>96</td>
</tr>
<tr>
<td>196.608</td>
<td>16 or 32</td>
<td>2</td>
<td>0</td>
<td>-</td>
<td>192</td>
</tr>
</tbody>
</table>

1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 55.3: SPI implementation to check the supported data size.

1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10). Refer to Section 55.3: SPI implementation to check the supported data size.
It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to Section 55.9.11 and Section 55.9.15 for additional information.

**FIFO size optimization**

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes 3 basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:
- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)
- 4 samples, if the DATLEN = 2 (32 bits)

**55.9.11 FIFOs status flags**

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

**TxFIFO threshold reached (TXP)**

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. thus FTHLV new data to be transmitted can be written into SPI_TXDR. The TXP flag is reset when the amount of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

**RxFIFO threshold reached (RXP)**

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See Section 55.10 for additional information on interrupt function in I2S mode.

**55.9.12 Handling of underrun situation**

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

---

*Not always available, refer to Section 55.3: SPI implementation in order to check if 24 and 32-bit data width are supported.*
For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral re-plays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the UDR flag is cleared by the software, and
- When the restart condition is met:
  - if the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - if the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

**Figure 804. Handling of underrun situation**

When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO, and the UDR flag is cleared.

The UDR flag can trigger an interrupt if the UDRIE bit in the SPI_IER register is set. The UDR bit is cleared by writing UDRC bit of SPIIFnCR register to 1.

**Note:** An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK and MCK signal are not gated.

Due to resynchronization, any change on the UDR flag is taken into account by the hardware after at least 2 periods of CK clock.

### 55.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.
In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO, and the OVR flag is cleared. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 805. Handling of overrun situation

When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, and the OVR flag is cleared, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by writing OVRC bit of SPI_IFCR register to 1.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK and MCK signal are not gated.

55.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

- A frame synchronization received while the shift-in or shift-out of the previous data is not completed (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurring at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.
Figure 806 shows an example of frame error detection. The SPI/I2S block is in slave mode and the amount of bit clock periods for left channel are not enough to shift-in or shift-out the data. The figure shows that the on-going transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 806. Frame error detection, with FIXCH = 0

An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in Figure 807, in this mode the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: Figure 806 and Figure 807 show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 807. Frame error detection, with FIXCH = 1

The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.
55.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of SPI_CFG1 register.

In receive mode, the DMA interface is working as follow:
1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follow:
1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

55.9.16 Programming examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.
Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set i2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master Full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example, if a threshold of 2 audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. Clear all interrupt enable fields located in SPI_IER register.
8. Clear all status flags, by setting SUSPC, TIFREC, UDRC, OVRC of SPI_IFCR register.
9. Set SPE bit, then the MCK is generated. In the example presented here, the TxFIFO is not filled-up as soon as SPE = 1. The application can then configure the external audio codec via an I2C interface even if audio samples to transmit are not yet available. An Alternative sequence is proposed hereafter.
10. When the application wants to start the data transfer:
   - If the data transfer uses DMA:
     - Program the DMA peripheral: two channels, one for RX and one for TX
     - Initialize the memory buffer with valid audio samples for TX path – Enable the DMA channels,
     - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
     - Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled-up.
   - If the data transfer is done via interrupt:
     - Enable the interrupt events UDRIE, OVRIE, TXPIE and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into SPI_CR1 register.

Alternative sequence

- Steps 1 to 8 similar to the previous sequence.
- If the data transfer uses DMA:
  - Program the DMA peripheral: two channels, one for RX and one for TX
  - Initialize the memory buffer with valid audio samples for TX path
  - Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
  - Enable the DMA channels,
- If the data transfer is done via interrupt:
– Enable the interrupt events UDRIE, OVRIE, TXPIE and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.

• Set SPE, as soon as this bit is set to one the following actions may happen:
  – If the interrupt generation is enabled, the SPI/I2S generates an interrupt request allowing the interrupt handler to fill-up the TxFIFO.
  – If the DMA transfer are enabled, the SPI/I2S generates DMA requests in order to fill-up the TxFIFO
  – The MCK is generated. The application can then configure the external audio codec via an I2C interface.

• Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into SPI_CR1 register.

Stop Procedure in master mode
1. Set the bit CSUSP, in order to stop on-going transfers
2. Check the value of CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure
1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold, by setting the wanted value into FTHLV. For example if a threshold of 2 audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   – Program the DMA peripheral: one RX channel
   – Enable the DMA channel,
   – In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

**Stop Procedure in slave mode**
1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

### 55.10 I2S interrupts

In PCM/I2S mode an interrupt (spi_it) or a wake-up event signal (spi_wkup) can be generated according to the events described in the Table 584.

Interrupt events can be enabled and disabled separately.

<table>
<thead>
<tr>
<th>Interrupt vector</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event/Interrupt clearing method</th>
<th>Exit Sleep mode</th>
<th>Exit Stop modes</th>
<th>Exit Standby mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>TxFIFO threshold reached</td>
<td>TXP</td>
<td>When the TxFIFO contains less than FTHLV empty locations</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>Rx FIFO threshold reached</td>
<td>RXP</td>
<td>When the RxFIFO contains less than FTHLV samples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Overrun error</td>
<td>OVR</td>
<td>Write OVRC to 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Underrun error</td>
<td>UDR</td>
<td>Write UDRC to 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Frame error flag</td>
<td>TIFRE</td>
<td>Write TIFREC to 1</td>
<td></td>
<td></td>
<td>No</td>
</tr>
</tbody>
</table>

### 55.11 SPI/I2S registers

#### 55.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x00
Reset value: 0x0000 0000

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|------|
| 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0   |
| rw | rw | rw | rw | rw | rw | rw | rw | rs | rw | rw | rw | rw | rw | rw | rw |
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **IOLOCK**: locking the AF configuration of associated I/Os
This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs
0: AF configuration is not locked
1: AF configuration is locked
When this bit is set, SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

Bit 15  **TCRCINI**: CRC calculation initialization pattern control for transmitter
0: all zero pattern is applied
1: all ones pattern is applied

Bit 14  **RCRCINI**: CRC calculation initialization pattern control for receiver
0: All zero pattern is applied
1: All ones pattern is applied

Bit 13  **CRC33_17**: 32-bit CRC polynomial configuration
0: Full size (33-bit or 17-bit) CRC polynomial is not used
1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12  **SSI**: internal SS signal input level
This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral SS input internally and the I/O value of the SS pin is ignored.

Bit 11  **HDDIR**: Rx/Tx direction at half-duplex mode
In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.
0: SPI is receiver
1: SPI is transmitter

Bit 10  **CSUSP**: master suspend request
This bit reads as zero.
In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transaction.
The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.
After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transaction starts.

Bit 9  **CSTART**: master transfer start
This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transaction suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 55.9.8: Stop sequence*.
0: master transfer is at idle
1: master transfer is ongoing or temporary suspended by automatic suspend
In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.
Bit 8 **MASRX**: master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)
1: SPI flow is suspended temporary on Rx FIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay. This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles should always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transaction without any next constraint. Prior the SUSP bit is cleared, the user must release the Rx FIFO space as much as possible by reading out all the data packets available at Rx FIFO based on the RXP flag indication to prevent any subsequent suspension.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE**: serial peripheral enable

This bit is set by and cleared by software.

0: Serial peripheral disabled.
1: Serial peripheral enabled

When SPE = 1, SPI data transfer is enabled, SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the SPI_clk clock), the SS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero. SPE is cleared and cannot be set when MODF error flag is active.

### 55.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Serial peripheral interface (SPI) RM0477

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **TSIZE[15:0]**: number of data at current transfer
   - When these bits are changed by software, the SPI must be disabled.
   - Endless transaction is initialized when CSTART is set while zero value is stored at TSIZE.
   - TSIZE cannot be set to 0xFFFF respective 0x3FFF value when CRC is enabled.
   - **Note**: TSIZE[15:10] bits are reserved at limited feature set instances and must be kept at reset value.

55.11.3  **SPI/I2S configuration register 1 (SPI_CFG1)**

Address offset: 0x08

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>BPASS</strong>: bypass of the prescaler at master baud rate clock generator</td>
<td>0: bypass is disabled, 1: bypass is enabled</td>
</tr>
<tr>
<td>30:28</td>
<td><strong>MBR[2:0]</strong>: master baud rate prescaler setting</td>
<td>000: SPI master clock/2, 001: SPI master clock/4, 010: SPI master clock/8, 011: SPI master clock/16, 100: SPI master clock/32, 101: SPI master clock/64, 110: SPI master clock/128, 111: SPI master clock/256</td>
</tr>
<tr>
<td>27:23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td><strong>CRCEN</strong>: hardware CRC computation enable</td>
<td>0: CRC calculation disabled, 1: CRC calculation enabled</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>
Bits 20:16 **CRCSIZE[4:0]**: length of CRC frame to be transacted and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transacted or compared. The length of the polynomial is not affected by this setting.

- 00000: reserved
- 00001: reserved
- 00010: reserved
- 00011: 4-bits
- 00100: 5-bits
- 00101: 6-bits
- 00110: 7-bits
- 00111: 8-bits
- ..... 11101: 30-bits
- 11110: 31-bits
- 11111: 32-bits
The value must be set equal or multiply of data size (DSIZE[4:0]). Its maximum size corresponds to DSIZE maximum at the instance.

*Note: The most significant bit at CRCSIZE bit field is reserved at the peripheral instances where data size is limited to 16-bit.*

Bit 15 **TXDMAEN**: Tx DMA stream enable
- 0: Tx DMA disabled
- 1: Tx DMA enabled

Bit 14 **RXDMAEN**: Rx DMA stream enable
- 0: Rx-DMA disabled
- 1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.
Bit 9  **UDRCFG**: behavior of slave transmitter at underrun condition

0: slave sends a constant pattern defined by the user at the SPI_UDRDR register
1: Slave repeats lastly received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

For more details see Figure 788: Optional configurations of the slave behavior when an underrun condition is detected.

Bits 8:5  **FTHLV[3:0]**: FIFO threshold level

Defines number of data frames in a single data packet. Size of the packet should not exceed 1/2 of FIFO space.

- 0000: 1-data
- 0001: 2-data
- 0010: 3-data
- 0011: 4-data
- 0100: 5-data
- 0101: 6-data
- 0110: 7-data
- 0111: 8-data
- 1000: 9-data
- 1001: 10-data
- 1010: 11-data
- 1011: 12-data
- 1100: 13-data
- 1101: 14-data
- 1110: 15-data
- 1111: 16-data

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

- If SPI data register is accessed as a 16-bit register and DSIZE \(\leq\) 8 bit, better to select FTHLV = 2, 4, 6.
- If SPI data register is accessed as a 32-bit register and DSIZE > 8 bit, better to select FTHLV = 2, 4, 6, while if DSIZE \(\leq\) 8bit, better to select FTHLV = 4, 8, 12.

Note:  **FTHLV[3:2]** bits are reserved at instances with limited set of features.
Bits 4:0 **DSIZE[4:0]**: number of bits in a single SPI data frame

- 00000: not used
- 00001: not used
- 00010: not used
- 00011: 4 bits
- 00100: 5 bits
- 00101: 6 bits
- 00110: 7 bits
- 00111: 8 bits
- ...
- 11101: 30 bits
- 11110: 31 bits
- 11111: 32 bits

**Note:** Maximum data size can be limited up to 16-bits at some instances. At instances with limited set of features, DSIZE[2:0] bits are reserved and must be kept at reset state. DSIZE[4:3] bits then control next settings of data size:

- 00xxx: 8-bits
- 01xxx: 16-bits
- 10xxx: 24-bits
- 11xxx: 32-bits.

### 55.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x0C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled or IOLOCK bit is set at SPI_CR1 register.
Bit 31  **AFCNTR**: alternate function GPIOs control  
This bit is taken into account when SPE = 0 only  
0: The peripheral takes no control of GPIOs while it is disabled  
1: The peripheral keeps always control of all associated GPIOs  
When SPI must be disabled temporarily for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.  
*Note*: This bit can be also used in PCM and I2S modes.  
*Note*: The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30  **SSOM**: SS output management in master mode  
This bit is taken into account in master mode when SSOE is enabled. It allows the SS output to be configured between two consecutive data transfers.  
0: SS is kept at active level till data transfer is completed, it becomes inactive with EOT flag  
1: SPI data frames are interleaved with SS non active pulses when MIDI[3:0]>1

Bit 29  **SSOE**: SS output enable  
This bit is taken into account in master mode only  
0: SS output is disabled and the SPI can work in multimaster configuration  
1: SS output is enabled. The SPI cannot work in a multimaster environment. It forces the SS pin at inactive level after the transfer is completed or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28  **SSIOP**: SS input/output polarity  
0: low level is active for SS signal  
1: high level is active for SS signal

Bit 27  Reserved, must be kept at reset value.

Bit 26  **SSM**: software management of SS signal input  
0: SS input value is determined by the SS PAD  
1: SS input value is determined by the SSI bit  
When master uses hardware SS output (SSM = 0 and SSOE = 1) the SS signal input is forced to not active state internally to prevent master mode fault error.

Bit 25  **CPOL**: clock polarity  
0: SCK signal is at 0 when idle  
1: SCK signal is at 1 when idle

Bit 24  **CPHA**: clock phase  
0: the first clock transition is the first data capture edge  
1: the second clock transition is the first data capture edge

Bit 23  **LSBFIRST**: data frame format  
0: MSB transmitted first  
1: LSB transmitted first  
*Note*: This bit can be also used in PCM and I2S modes.

Bit 22  **MASTER**: SPI master  
0: SPI slave  
1: SPI master
Bits 21:19 **SP[2:0]**: serial protocol  
- 000: SPI Motorola  
- 001: SPI TI  
- others: reserved, must not be used  

Bits 18:17 **COMM[1:0]**: SPI Communication Mode  
- 00: full-duplex  
- 01: simplex transmitter  
- 10: simplex receiver  
- 11: half-duplex  

Bit 16 Reserved, must be kept at reset value.  

Bit 15 **IOSWP**: swap functionality of MISO and MOSI pins  
- 0: no swap  
- 1: MOSI and MISO are swapped  

When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.  

*Note:* This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.  

Bit 14 **RDIOP**: RDY signal input/output polarity  
- 0: high level of the signal means the slave is ready for communication  
- 1: low level of the signal means the slave is ready for communication  

Bit 13 **RDIOM**: RDY signal input/output management  
- 0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)  
- 1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)  

*Note:* When DSIZE at the SPI_CFG1 register is configured shorter than 8-bit, the RDIOM bit must be kept at zero.  

Bits 12:8 Reserved, must be kept at reset value.  

Bits 7:4 **MIDI[3:0]**: master Inter-Data Idleness  
Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.  
- 0000: no delay  
- 0001: 1 clock cycle period delay  
- ...  
- 1111: 15 clock cycle periods delay  

*Note:* This feature is not supported in TI mode.  

Bits 3:0 **MSSI[3:0]**: Master SS Idleness  
Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of SS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.  
- 0000: no extra delay  
- 0001: 1 clock cycle period delay added  
- ...  
- 1111: 15 clock cycle periods delay added  

*Note:* This feature is not supported in TI mode.  
*To include the delay, the SPI must be disabled and re-enabled between sessions.*
55.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x10
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 9 **MODFIE**: mode Fault interrupt enable
0: MODF interrupt disabled
1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable
0: TIFRE interrupt disabled
1: TIFRE interrupt enabled

Bit 7 **CRCEIE**: CRC error interrupt enable
0: CRC interrupt disabled
1: CRC interrupt enabled

Bit 6 **OVRIE**: OVR interrupt enable
0: OVR interrupt disabled
1: OVR interrupt enabled

Bit 5 **UDRIE**: UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE**: TXTFIE interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE**: EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE**: TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE**: RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled
55.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x14
Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is re-enabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

<table>
<thead>
<tr>
<th>Bits 31:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTSIZE[15:0]: number of data frames remaining in current TSIZE session</td>
</tr>
<tr>
<td>The value is not quite reliable when traffic is ongoing on bus.</td>
</tr>
<tr>
<td><strong>Note:</strong> CTSIZE[15:0] bits are not available in instances with limited set of features.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXWNE: RxFIFO word not empty</td>
</tr>
<tr>
<td>0: less than four bytes of RxFIFO space is occupied by data</td>
</tr>
<tr>
<td>1: at least four bytes of RxFIFO space is occupied by data</td>
</tr>
<tr>
<td><strong>Note:</strong> This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 14:13</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXPLVL[1:0]: RxFIFO packing level</td>
</tr>
<tr>
<td>When RXWNE = 0 and data size is set up to 16-bit, the value gives number of remaining data frames persisting at RxFIFO.</td>
</tr>
<tr>
<td>00: no next frame is available at RxFIFO</td>
</tr>
<tr>
<td>01: 1 frame is available</td>
</tr>
<tr>
<td>10: 2 frames are available*</td>
</tr>
<tr>
<td>11: 3 frames are available*</td>
</tr>
<tr>
<td><strong>Note:</strong> (*): Possible value when data size is set up to 8-bit only.</td>
</tr>
<tr>
<td>When data size is greater than 16-bit, these bits are always read as 00. In that consequence, the single data frame received at the FIFO cannot be detected neither by RXWNE nor by RXPLVL bits if data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE &gt; 0 or RXP events when FTHLV = 0.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXC: TxFIFO transmission complete</td>
</tr>
<tr>
<td>The flag behavior depends on TSIZE setting.</td>
</tr>
<tr>
<td>When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.</td>
</tr>
<tr>
<td>If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.</td>
</tr>
<tr>
<td>This flag is set when SPI is reset or disabled.</td>
</tr>
<tr>
<td>0: current data transaction is still ongoing, data is available in TxFIFO or last frame transmission is on going.</td>
</tr>
<tr>
<td>1: last TxFIFO frame transmission complete</td>
</tr>
</tbody>
</table>

---

[Table and bit descriptions]
Bit 11 **SUSP**: suspension status
   In master mode, SUSP is set by hardware either as soon as the current frame is completed after CSUSP request is done or at master automatic suspend receive mode (MASRX bit is set at SPI_CR1 register) on RxFIFO full condition. SUSP generates an interrupt when EOTIE is set. This bit must be cleared prior SPI is disabled and this is done by writing 1 to SUSPC bit of SPI_IFCR exclusively.
   0: SPI not suspended (master mode active or other mode).
   1: Master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault
   0: no mode fault
   1: mode fault detected.
   When MODF is set, SPE and IOLOCK bits of SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.
   This bit is cleared by writing 1 to MODFC bit of SPI_IFCR exclusively.

Bit 8 **TIFRE**: TI frame format error
   0: no TI Frame Error
   1: TI frame error detected
   This bit is cleared by writing 1 to TIFREC bit of SPI_IFCR exclusively.

Bit 7 **CRCE**: CRC error
   0: no CRC error
   1: CRC error detected
   This bit is cleared when SPI is re-enabled or by writing 1 to CRCEC bit of SPI_IFCR optionally.

Bit 6 **OVR**: overrun
   0: no overrun
   1: overrun detected
   This bit is cleared when SPI is re-enabled or by writing 1 to OVRC bit of SPI_IFCR optionally.

Bit 5 **UDR**: underrun
   0: no underrun
   1: underrun detected
   This bit is cleared when SPI is re-enabled or by writing 1 to UDRC bit of SPI_IFCR optionally.

*Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.*

Bit 4 **TXTF**: transmission transfer filled
   0: upload of TxFIFO is ongoing or not started
   1: TxFIFO upload is finished
   TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.
   This bit is cleared by software write 1 to TXTFC bit of SPI_IFCR exclusively.
   TXTF flag triggers an interrupt if TXTFIE bit is set.
   TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.
Bit 3  **EOT**: end of transfer  
EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to EOT bit of SPI_IFCR optionally.  
EOT flag triggers an interrupt if EOTIE bit is set.  
If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.  
0: transfer is ongoing or not started  
1: transfer complete  
In master, EOT event terminates the data transaction and handles SS output optionally.  
When CRC is applied, the EOT event is extended over the CRC frame transaction.  
To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transaction starts despite its setting is not changed.

Bit 2  **DXP**: duplex packet  
0: Tx FIFO is Full and/or Rx FIFO is Empty  
1: both Tx FIFO has space for write and Rx FIFO contains for read a single packet at least  
DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

Bit 1  **TXP**: Tx-packet space available  
0: not enough free space at Tx FIFO to host next data packet  
1: enough free space at Tx FIFO to host at least one data packet  
TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at Tx FIFO. TXP is set despite SPI Tx FIFO becomes inaccessible when SPI is reset or disabled.

Bit 0  **RXP**: Rx-packet available  
0: Rx FIFO is empty or an incomplete data packet is received  
1: Rx FIFO contains at least one data packet  
The flag is changed by hardware. It monitors the total number of data currently available at Rx FIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from Rx FIFO.

### 55.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x18  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9  | Bit 8  | Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| w      | w      | w      | w      | w      | w      | w      | w      | w      | w      | w      | w      | w      | w      | w      | w      |

RM0477 Rev 6  
2625/3791
Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC**: Suspend flag clear
- Writing a 1 into this bit clears SUSP flag in the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear
- Writing a 1 into this bit clears MODF flag in the SPI_SR register

Bit 8 **TIFREC**: T1 frame format error flag clear
- Writing a 1 into this bit clears TIFRE flag in the SPI_SR register

Bit 7 **CRCEC**: CRC error flag clear
- Writing a 1 into this bit clears CRCE flag in the SPI_SR register

Bit 6 **OVR**: overrun flag clear
- Writing a 1 into this bit clears OVR flag in the SPI_SR register

Bit 5 **UDRC**: underrun flag clear
- Writing a 1 into this bit clears UDR flag in the SPI_SR register

Bit 4 **TXTFC**: transmission transfer filled flag clear
- Writing a 1 into this bit clears TXTF flag in the SPI_SR register

Bit 3 **EOTC**: end of transfer flag clear
- Writing a 1 into this bit clears EOT flag in the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

### 55.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Bit 15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXDR[31:16]</td>
<td>TXDR[15:0]</td>
</tr>
<tr>
<td>w w w w w w w w w w w w w w w</td>
<td>w w w w w w w w w w w w w w w</td>
</tr>
</tbody>
</table>

Bits 31:0 **TXDR[31:0]**: transmit data register
- The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: **In SPI mode, data is always right-aligned.** Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: **DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.**
- half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.
- word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.
55.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x30
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

RXDR[31:16]

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
</tbody>
</table>

RXDR[15:0]

Bits 31:0 RXDR[31:0]: receive data register

The register serves as an interface with Rx_FIFO. When it is read, Rx_FIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read.
Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access
half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access
word-wise (32 bit access), In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.
Read access of this register less than the configured data size is forbidden.

55.11.10 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x40
Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

CRCPOLY[31:16]

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

CRCPOLY[15:0]
55.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x44
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>TXCRC[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 5 1 4 1 3 1 2 1 1 1 0 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>TXCRC[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>r r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of SPI_CR1 is set or when a data block is transacted completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on SPI_CRCPOLY register and CRCSIZE bits settings at SPI_CFG1 register.

*Note:* A read to this register when the communication is ongoing may return an incorrect value.

*Note:* This bitfield is not used in I2S mode.

*Note:* TXCRC[31-16] bits are reserved at instances with data size limited to 16-bit. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

*Note:* The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.
55.11.12  SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x48
Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

55.11.13  SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x4C
Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.
55.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x50

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit which needs to be kept at 0.

This register is reserved at instances not supporting I2S mode.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Address offset: 0x50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset value: 0x0000 0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit which needs to be kept at 0.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>This register is reserved at instances not supporting I2S mode.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 UDRDR[31:0]: data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on DSIZE bit settings of the SPI_CFG1 register. Underrun condition handling depends on setting UDRCFG bit at SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved at the peripheral instances with data size limited to 16-bit. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

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

- **Bit 25** MCKOE: master clock output enable
  - 0: Master clock output is disabled
  - 1: Master clock output is enabled

- **Bit 24** ODD: odd factor for the prescaler
  - 0: Real divider value is = I2SDIV * 2
  - 1: Real divider value is = (I2SDIV * 2) + 1
  - Refer to Section 55.9.9: Clock generator for details

Bits 23:16 I2SDIV[7:0]: I2S linear prescaler

- I2SDIV can take any values except the value 1, when ODD is also equal to 1.
- Refer to Section 55.9.9: Clock generator for details

- **Bit 15** Reserved, must be kept at reset value.

- **Bit 14** DATFMT: data format
  - 0: The data inside the SPI_RXDR or SPI_TXDR are right aligned
  - 1: The data inside the SPI_RXDR or SPI_TXDR are left aligned.
Bit 13 **WSINV**: word select inversion  
This bit is used to invert the default polarity of WS signal.  
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.  
In MSB or LSB justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.  
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.  
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.  
In MSB or LSB justified mode, the left channel is transferred when WS is LOW, and right channel when WS is HIGH.  
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave  
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)  
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity  
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.  
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)  
0: 16-bit wide  
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.  
00: 16-bit data length  
01: 24-bit data length  
10: 32-bit data length  
11: Not allowed  
*Note*: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 55.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization  
0: short frame synchronization  
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I2S standard selection  
00: I2S Philips standard.  
01: MSB justified standard (left justified)  
10: LSB justified standard (right justified)  
11: PCM standard  
For more details on I2S standards, refer to Section 55.9.5: Supported audio protocols
55.11.15 SPI/I2S register map

### Table 585. SPI register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 3:1 I2SCFG[2:0]: I2S configuration mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SPI_CR1</td>
<td>000: slave - transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: slave - receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: master - transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: master - receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: slave - Full Duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: master - Full Duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>others, not used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0 I2SMOD: I2S mode selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: SPI mode is selected</td>
</tr>
<tr>
<td>1: I2S/PCM mode is selected</td>
</tr>
</tbody>
</table>

#### SPI/I2S register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 3:1 I2SCFG[2:0]: I2S configuration mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SPI_CR1</td>
<td>000: slave - transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: slave - receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: master - transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: master - receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: slave - Full Duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: master - Full Duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>others, not used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0 I2SMOD: I2S mode selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: SPI mode is selected</td>
</tr>
<tr>
<td>1: I2S/PCM mode is selected</td>
</tr>
</tbody>
</table>

#### SPI register map

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits 3:1 I2SCFG[2:0]: I2S configuration mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>SPI_CR1</td>
<td>000: slave - transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: slave - receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: master - transmit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: master - receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: slave - Full Duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: master - Full Duplex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>others, not used</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0 I2SMOD: I2S mode selection</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: SPI mode is selected</td>
</tr>
<tr>
<td>1: I2S/PCM mode is selected</td>
</tr>
</tbody>
</table>
### Table 585. SPI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x44</td>
<td>SPI_TXCRC</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>TXCRC<a href="2">31:16</a> TXCRC[16:0]</td>
</tr>
<tr>
<td></td>
<td>SPI_RXCRC</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>RXCRC<a href="2">31:16</a> RXCRC[16:0]</td>
</tr>
<tr>
<td></td>
<td>SPI_UDRDR</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>UDRDR<a href="2">31:16</a> UDRDR[16:0]</td>
</tr>
<tr>
<td>0x50</td>
<td>SPI_I2SCFGR</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>MCKOE ODD I2SDIV(7:0) WSTINV FIXCH CKPOL CHLEN DAREN(1:0) POMSYNC IOSLOT[7:0] I2SCFG2[2:0] I2SMOD</td>
</tr>
</tbody>
</table>

1. The bitfield is reserved for instances with limited set of features and it must be kept at reset value. For more details, refer to the concrete register description in Section 55.11: SPI/I2S registers.

2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16-bit. There is no constrain when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to Section 2.3 for the register boundary addresses.
56 Serial audio interface (SAI)

56.1 Introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many stereo or mono audio applications may be targeted. I2S standards, LSB or MSB-justified, PCM/DSP, TDM, and AC’97 protocols may be addressed for example. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has its own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

56.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Possible synchronization between multiple SAIs
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception
• Interrupt sources when enabled:
  – Errors
  – FIFO requests
• 2-channel DMA interface

56.3 SAI implementation

<table>
<thead>
<tr>
<th>SAI features</th>
<th>SAI1</th>
<th>SAI2</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2S, LSB or MSB-justified, PCM/DSP, TDM, AC’97</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>FIFO size</td>
<td>8 words</td>
<td>8 words</td>
</tr>
<tr>
<td>SPDIF</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PDM</td>
<td>X(2)</td>
<td>-</td>
</tr>
</tbody>
</table>

1. ‘X’ = supported, ‘-’ = not supported.

56.4 SAI functional description

56.4.1 SAI block diagram

*Figure 808* shows the SAI block diagram while *Table 587* and *Table 588* list SAI internal and external signals.
The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/ Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC’97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).
Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where ‘x’ represents the SAI A or B subblock.

56.4.2 SAI pins and internal signals

### Table 587. SAI internal input/output signals

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sai_a_gbl_it/sai_b_gbl_it</td>
<td>Output</td>
<td>Audio block A and B global interrupts</td>
</tr>
<tr>
<td>sai_a_dma/sai_b_dma</td>
<td>Input/output</td>
<td>Audio block A and B DMA acknowledges and requests</td>
</tr>
<tr>
<td>sai_sync_out_sck/sai_sync_out_fs</td>
<td>Output</td>
<td>Internal clock and frame synchronization output signals exchanged with other SAI blocks</td>
</tr>
<tr>
<td>sai_sync_in_sck/sai_sync_in_fs</td>
<td>Input</td>
<td>Internal clock and frame synchronization input signals exchanged with other SAI blocks</td>
</tr>
<tr>
<td>sai_a_ker_ck/sai_b_ker_ck</td>
<td>Input</td>
<td>Audio block A/B kernel clock</td>
</tr>
<tr>
<td>sai_pclk</td>
<td>Input</td>
<td>APB clock</td>
</tr>
</tbody>
</table>

### Table 588. SAI input/output pins

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Pin type</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAI_SCK_A/B</td>
<td>Input/output</td>
<td>Audio block A/B bit clock</td>
</tr>
<tr>
<td>SAI_MCLK_A/B</td>
<td>Output</td>
<td>Audio block A/B master clock</td>
</tr>
<tr>
<td>SAI_SD_A/B</td>
<td>Input/output</td>
<td>Data line for block A/B</td>
</tr>
<tr>
<td>SAI_FS_A/B</td>
<td>Input/output</td>
<td>Frame synchronization line for audio block A/B</td>
</tr>
<tr>
<td>SAI_CK[2:1]</td>
<td>Output</td>
<td>PDM bitstream clock(^{(1)})</td>
</tr>
<tr>
<td>SAI_D[3:1]</td>
<td>Input</td>
<td>PDM bitstream data(^{(1)})</td>
</tr>
</tbody>
</table>

1. These signals might not be available in all SAI instances. Refer to Section 56.3: SAI implementation for details.

56.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

**Master mode**

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.
Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:
1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold (FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

56.4.4 SAI synchronization mode

There are two levels of synchronization, either at audio subblock level or at SAI level.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).
Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding \text{SYNCEN}[1:0] bits set to 00 in SAI\textunderscore xCR1) and one synchronous block (corresponding \text{SYNCEN}[1:0] bits set to 01 in the SAI\textunderscore xCR1 register).

\textbf{Note:} Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

\textbf{External synchronization}

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follows:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblocks is supposed to provide the FS and SCK signals to other SAI. This is done by programming \text{SYNCOUT}[1:0] bits.

2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on \text{SYNCIN}[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the \text{SYNCEN} bit.

\textbf{Note:} The \text{SYNCIN}[1:0] and \text{SYNCOUT}[1:0] bits are located in the SAI\textunderscore GCR register, and the \text{SYNCEN} bits in the SAI\textunderscore xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the \text{SYNCEN}[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the \text{SYNCEN}[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through \text{SYNCEN}[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as ‘Reserved’ must not be used.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
\textbf{Block instance} & \textbf{SYNCIN = 1} & \textbf{SYNCIN = 0} \\
\hline
SAI1 & SAI2 sync. & Reserved \\
SAI2 & Reserved & SAI1 sync. \\
\hline
\end{tabular}
\end{table}

\textbf{56.4.5 Audio data size}

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI\textunderscore xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI\textunderscore xCR1 register.
56.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. Figure 809 illustrates this flexibility.

Figure 809. Audio frame

In AC’97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC’97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode
  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z depending on the state of bit TRIS in the SAI_xCR2 register (refer to FS signal role). In reception mode, the remaining bit is ignored.

  If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

  If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to Section 56.4.8: SAI clock generator.

- Slave mode
  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to Section 56.4.14: Error flags.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

  The number of bits in the frame is equal to FRL[7:0] + 1.

  The minimum number of bits to transfer in an audio frame is 8.
Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to Section 56.4.14: Error flags).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in Section 56.4.14: Error flags, but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC’97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.
If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.
The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

### 56.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

For AC’97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to Output data line management on an inactive slot) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

The slot size is also configurable as shown in Figure 812. The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTTR register. The size is applied identically for each slot in an audio frame.
Figure 812. Slot size configuration with FBOFF = 0 in SAI_xSLOTTR

It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 813. First bit offset

It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤ (SLOTSZ - DS)
- DS ≤ SLOTSZ
- NBSLOT × SLOTSZ ≤ FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC’97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFCG[1:0] = 01), the slot size is automatically set as defined in Section 56.4.11: AC’97 link controller.
56.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

**Generation of the master clock (MCLK_x)**

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:
- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

*Table 590* shows MCLK_x activation conditions.

**Table 590. MCLK_x activation conditions**

<table>
<thead>
<tr>
<th>MCLKEN</th>
<th>NODIV</th>
<th>SAIEN for block x</th>
<th>MCLK_x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>Enabled</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td></td>
<td>Enabled</td>
</tr>
</tbody>
</table>

*Note:* MCLK_x can also be generated in AC’97 mode, when MCLKEN is set to 1.
**Generation of the bit clock (SCK\_x)**

The clock generator provides the bit clock (SCK\_x) when the audio block is defined as Master. The frame synchronization (FS\_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK\_x clock is not generated.

The bit clock strobing edge of SCK\_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

*Figure 814* illustrates the architecture of the audio block clock generator.

---

**Figure 814. Audio block clock generator overview**

The NODIV bit must be used to force the ratio between the master clock (MCLK\_x) and the frame synchronization (FS\_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK\_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] \(\geq\) 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK\_x is equal to the SCK\_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI\_xCR1 register, while FRL[7:0] belongs to SAI\_xFRCR.
Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

• $F_{MCLK_x} = 256 \times F_{FS_x}$ if OSR = 0
• $F_{MCLK_x} = 512 \times F_{FS_x}$ if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK_x} = \frac{F_{sai\_x\_ker\_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = $2^8$.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai\_x\_ker\_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) is given by the following formula:

$$F_{FS_x} = \frac{F_{sai\_x\_ker\_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.
Clock generator programming examples

*Table 591* gives programming examples for 48, 96 and 192 kHz.

### Table 591. Clock generator programming examples

<table>
<thead>
<tr>
<th>Input sai_x_ker_ck clock frequency</th>
<th>MCLK</th>
<th>$F_{MCLK}/F_{FS}$</th>
<th>FRL (i)</th>
<th>OSR</th>
<th>NODIV</th>
<th>MCKEN</th>
<th>MCKDIV[5:0]</th>
<th>Audio Sampling frequency ($F_{FS}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>98.304 MHz</td>
<td>512 $2^{N-1}$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 or 1</td>
<td>8</td>
<td>192 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512 $2^{N-1}$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>96 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>512 $2^{N-1}$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>48 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>256 $2^{N-1}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>192 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>256 $2^{N-1}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>96 kHz</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td>256 $2^{N-1}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
<td>48 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 63</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>192 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 63</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>16</td>
<td>96 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 63</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>32</td>
<td>48 kHz</td>
<td></td>
</tr>
</tbody>
</table>

1. N is an integer value between 3 and 8.

### 56.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

#### Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).
When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).

When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

• When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.

• When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).

• When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).

• When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).

• When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This
interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

56.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to Section 56.3: SAI implementation).

Figure 815 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the TDM interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).
The data processing sequence into the PDM is the following:
1. The PDM interface builds the bitstream clock from the bit clock received from the TDM interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A via the serial data line of the TDM interface.

Figure 8.16 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

![Figure 8.16. Detailed PDM interface block diagram](image)

1. \( n \) refers to the number of data lines and \( p \) to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to Section 56.3: SAI implementation for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.
To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

**Enabling the PDM interface**

To enable the PDM interface, follow the sequence below:

1. Configure SAI_A in TDM master mode (see Table 592).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

**Note:** Once the PDM interface and SAI_A are enabled, the first two TDMA frames received on SAI_ADR are invalid and must be dropped.

**Startup sequence**

*Figure 817* shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the TDM interface.
SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 818*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

*Figure 818. SAI_ADR format in TDM, 32-bit slot width*

- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 819*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.
• **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 820*. For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.
Figure 820. SAI_ADR format in TDM, 8-bit slot width

TDM configuration for PDM interface

SAI_A TDM interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the PDM interface as shown in Table 592 to ensure a good connection with the PDM interface.

Table 592. TDM settings

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODE</td>
<td>0b01</td>
<td>Mode must be MASTER receiver.</td>
</tr>
<tr>
<td>PRTCFG</td>
<td>0b00</td>
<td>Free protocol for TDM.</td>
</tr>
<tr>
<td>DS</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See Table 593.</td>
</tr>
<tr>
<td>LSBFIRST</td>
<td>X</td>
<td>This parameter can be used according to the desired data format.</td>
</tr>
<tr>
<td>CKSTR</td>
<td>0</td>
<td>Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.</td>
</tr>
<tr>
<td>MONO</td>
<td>0</td>
<td>Stereo mode.</td>
</tr>
<tr>
<td>FRL</td>
<td>X</td>
<td>To be adjusted according to the number of microphones (MICNBR). See Table 593.</td>
</tr>
<tr>
<td>FSALL</td>
<td>0</td>
<td>Pulse width is one bit clock cycle.</td>
</tr>
</tbody>
</table>
Adjusting the bitstream clock rate

To program the SAI TDM interface properly, the user application must take into account the settings given in Table 592, and follow the sequence below:

1. Adjust the bit clock frequency ($F_{SCK_A}$) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

MICNBR can be 0, 1, 2 or 3 ($0 = 2$ microphones; see Section 56.6.18)

2. Set the frame length (FRL) using the following formula

$$FRL = (16 \times (MICNBR + 1)) - 1$$

3. Configure the slot size (DS) to a multiple of (FRL+1).

---

### Table 592. TDM settings (continued)

<table>
<thead>
<tr>
<th>Bit Fields</th>
<th>Values</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSDEF</td>
<td>0</td>
<td>FS signal is a start of frame.</td>
</tr>
<tr>
<td>FSPOL</td>
<td>1</td>
<td>FS is active high.</td>
</tr>
<tr>
<td>FSOFF</td>
<td>0</td>
<td>FS is asserted on the first bit of slot 0.</td>
</tr>
<tr>
<td>FBOFF</td>
<td>0</td>
<td>No offset on slot.</td>
</tr>
<tr>
<td>SLOTSZ</td>
<td>0</td>
<td>Slot size = data size.</td>
</tr>
<tr>
<td>NBSLOT</td>
<td>X</td>
<td>To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See Table 593.</td>
</tr>
<tr>
<td>SLOTEN</td>
<td>X</td>
<td>To be adjusted according to NBSLOT.</td>
</tr>
<tr>
<td>NODIV</td>
<td>1</td>
<td>No need to generate a master clock MCLK.</td>
</tr>
<tr>
<td>MCKDIV</td>
<td>X</td>
<td>Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See Table 593.</td>
</tr>
</tbody>
</table>
Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two TDM frames.
56.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

*Figure 821* shows an AC’97 audio frame structure.

---

**Figure 821. AC’97 audio frame**

<table>
<thead>
<tr>
<th>FS</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDI</td>
<td>Tag</td>
<td>CMD ADDR</td>
<td>CMD DATA</td>
<td>POM FRONT</td>
<td>POM RFRONT</td>
<td>LINE1 DAC</td>
<td>LINE2 DAC</td>
<td>HSET DAC</td>
<td>IO CTRL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SDO</td>
<td>Tag</td>
<td>STATUS ADDR</td>
<td>STATUS DATA</td>
<td>POM LEFT</td>
<td>POM RIGHT</td>
<td>PCM LINE1 DAC</td>
<td>PCM LINE2 DAC</td>
<td>PCM HSET</td>
<td>PCM IO STATUS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note:* In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 822*.

In SAI1, the audio block A must be declared as asynchronous master transmitter, whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.
In receiver mode, the SAI acting as an AC’97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC’97 protocol.

Clock generator programming in AC’97 mode

In AC’97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in Section 56.4.8: SAI clock generator must be used with FRL = 255, to generate the proper frame rate ($F_{FS,x}$).
56.4.12 SPDIF output

The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958. To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:
- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in Figure 823.

Figure 823. SPDIF format

An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see Table 594). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 594. SOPD pattern

<table>
<thead>
<tr>
<th>SOPD</th>
<th>Preamble coding</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>last bit is 0</td>
<td>last bit is 1</td>
</tr>
<tr>
<td>B</td>
<td>111010000</td>
<td>00010111</td>
</tr>
<tr>
<td>W</td>
<td>111001000</td>
<td>00011011</td>
</tr>
<tr>
<td>M</td>
<td>111000010</td>
<td>00011101</td>
</tr>
</tbody>
</table>
The data stored in SAI_xDR has to be filled as follows:

- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].
If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].
SAI_xDR[23] always represents the MSB.

**Figure 824. SAI_xDR register ordering**

Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in Table 595.

**Table 595. Parity bit calculation**

<table>
<thead>
<tr>
<th>SAI_xDR[26:0]</th>
<th>Parity bit P value transferred</th>
</tr>
</thead>
<tbody>
<tr>
<td>odd number of 0</td>
<td>0</td>
</tr>
<tr>
<td>odd number of 1</td>
<td>1</td>
</tr>
</tbody>
</table>

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.
   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIENT bit in the SAI_xCR1 register.
Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

### Table 596. Audio sampling frequency versus symbol rates

<table>
<thead>
<tr>
<th>Audio sampling frequencies ($F_s$)</th>
<th>Symbol-rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>44.1 kHz</td>
<td>2.8224 MHz</td>
</tr>
<tr>
<td>48 kHz</td>
<td>3.072 MHz</td>
</tr>
<tr>
<td>96 kHz</td>
<td>6.144 MHz</td>
</tr>
<tr>
<td>192 kHz</td>
<td>12.288 MHz</td>
</tr>
</tbody>
</table>

More generally, the relationship between the audio sampling frequency ($F_s$) and the bit clock rate ($F_{SCK,x}$) is given by the formula:

$$F_s = \frac{F_{SCK,x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK,x} = \frac{F_{sai,x\_ker\_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

56.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

**Mute mode**

The mute mode can be used when the audio subblock is a transmitter or a receiver.

**Audio subblock in transmission mode**

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTTr register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTTr register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.
The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

**Audio subblock in reception mode**

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

*Note:* The mute mode is not available for SPDIF audio blocks.

**Mono/stereo mode**

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:
1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

**Companding mode**

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in Figure 825. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1’s complement or 2’s complement representation, depending on the CPL bit setting in the SAI_xCR2 register.
In μ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

**Figure 825. Data companding hardware in an audio block in the SAI**

1. Not applicable when AC’97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:
- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

**Output data line management on an inactive slot**

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame. **Figure 826** illustrates these behaviors.
Figure 826. Tristate strategy on SD output line on an inactive slot

When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to Figure 827 (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).
If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in Figure 826 and Figure 827 are replaced by a drive with a value of 0.

56.4.14 Error flags

The SAI implements the following error flags:
- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC’97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDR:interrupt bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to Figure 828).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to Figure 829). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 829. FIFO underrun event
Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in Anticipated frame synchronization detection (AFSDET)).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave. It has no meaning in SPDIF mode since the signal FS is not used by the protocol.
Codec not ready (CNRDY AC’97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC’97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC’97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

56.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

56.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to Section 56.4.9: Internal FIFOs). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.
Follow the sequence below to configure the SAI interface in DMA mode:
1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

### 56.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in Table 597.

<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt source</th>
<th>Interrupt group</th>
<th>Audio block mode</th>
<th>Interrupt enable</th>
<th>Interrupt clear</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQ</td>
<td>FREQ</td>
<td>FREQ</td>
<td>Master or slave</td>
<td>FREQIE in</td>
<td>Depends on:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Receiver or</td>
<td>SAI_xIM register</td>
<td>– FIFO threshold setting (FLVL bits in SAI_xCR2)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>transmitter</td>
<td></td>
<td>– Communication direction (transmitter or receiver)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>For more details refer to Section 56.4.9: Internal FIFOs</td>
</tr>
<tr>
<td>OVRUDR</td>
<td>ERROR</td>
<td>Master or slave</td>
<td>Receiver or</td>
<td>OVRUDRIE in</td>
<td>COVRUDR = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>transmitter</td>
<td>transmitter</td>
<td>SAI_xIM register</td>
<td></td>
</tr>
<tr>
<td>SAI</td>
<td>AFSDET</td>
<td>ERROR</td>
<td>Slave (not used in AC’97 mode and SPDIF mode)</td>
<td>AFSDETIE in</td>
<td>CAFSDET = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAI_xIM register</td>
<td></td>
</tr>
<tr>
<td>LFSDET</td>
<td>ERROR</td>
<td>Slave (not used in AC’97 mode and SPDIF mode)</td>
<td>LFSDETIE in</td>
<td>CLFSDET = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAI_xIM register</td>
<td></td>
</tr>
<tr>
<td>CNRDY</td>
<td>ERROR</td>
<td>Slave (only in AC’97 mode)</td>
<td>CNRDYIE in</td>
<td>CCNRDY = 1 in SAI_xCLRFR register</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAI_xIM register</td>
<td></td>
</tr>
<tr>
<td>MUTEDET</td>
<td>MUTE</td>
<td>Master or slave</td>
<td>Receiver mode only</td>
<td>MUTEDETIE in</td>
<td>CMUTEDET = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiver</td>
<td></td>
<td>SAI_xIM register</td>
<td></td>
</tr>
<tr>
<td></td>
<td>WCKCFG</td>
<td>ERROR</td>
<td>Master with NODIV = 0 in SAI_xCR1 register</td>
<td>WCKCFGIE in</td>
<td>CWCKCFG = 1 in SAI_xCLRFR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SAI_xIM register</td>
<td></td>
</tr>
</tbody>
</table>
Follow the sequence below to enable an interrupt:
1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.
56.6  SAI registers

The peripheral registers have to be accessed by words (32 bits).

56.6.1  SAI global configuration register (SAI_GCR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Reserved, must be kept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-0</td>
<td>SYNCOUT[1:0]</td>
<td>00</td>
<td>Synchronization outputs. 00: No synchronization output signals. SYNCOUT[1:0] must be configured as &quot;No synchronization output signals&quot; when audio block is configured as SPDIF. 01: Block A used for further synchronization for others SAI. 10: Block B used for further synchronization for others SAI. 11: Reserved. These bits must be set when both audio block (A and B) are disabled.</td>
</tr>
</tbody>
</table>

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0  SYNCIN[1:0]: Synchronization inputs. These bits are set and cleared by software. Refer to Table 589: External synchronization selection for information on how to program this field. They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

56.6.2  SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04
Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Default Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>Reserved, must be kept</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15-0</td>
<td>OUTDIR, MONO, SYNCEN[1:0], CKSTR, LSBFIRST, DS[2:0]</td>
<td>00000010 00000000</td>
<td>These bits are set and cleared by software. They must be set when both audio blocks (A and B) are disabled.</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.
Bit 27  **MCKEN**: Master clock generation enable
- 0: The master clock is not generated
- 1: The master clock is generated independently of SAIEN bit

Bit 26  **OSR**: Oversampling ratio for master clock
- This bit is meaningful only when NODIV bit is set to 0.
- 0: Master clock frequency = $F_{FS} \times 256$
- 1: Master clock frequency = $F_{FS} \times 512$

Bits 25:20  **MCKDIV[5:0]**: Master clock divider
- These bits are set and cleared by software.
- 000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
- Otherwise, the master clock frequency is calculated according to the formula given in Section 56.4.8: SAI clock generator.
- These bits have no meaning when the audio block is slave.
- They have to be configured when the audio block is disabled.

Bit 19  **NODIV**: No divider
- This bit is set and cleared by software.
- 0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
- 1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18  Reserved, must be kept at reset value.

Bit 17  **DMAEN**: DMA enable
- This bit is set and cleared by software.
- 0: DMA disabled
- 1: DMA enabled

  **Note**: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16  **SAIEN**: Audio block enable
- This bit is set by software.
- To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
- This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
- 0: SAI audio block disabled
- 1: SAI audio block enabled.

  **Note**: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

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

Bit 13  **OUTDRIV**: Output drive
- This bit is set and cleared by software.
- 0: Audio block output driven when SAIEN is set
- 1: Audio block output driven immediately after the setting of this bit.

  **Note**: This bit has to be set before enabling the audio block and after the audio block configuration.
Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot 1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable
These bits are set and cleared by software. They must be configured when the audio subblock is disabled.
00: audio subblock in asynchronous mode.
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode.
10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.
11: Reserved

*Note*: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR**: Clock strobing edge
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.
0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.
0: Data are transferred with MSB first
1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]**: Data size
These bits are set and cleared by software. These bits are ignored when the SPdif protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm. These bits must be configured when the audio block is disabled.
000: Reserved
001: Reserved
010: 8 bits
011: 10 bits
100: 16 bits
101: 20 bits
110: 24 bits
111: 32 bits

Bit 4 Reserved, must be kept at reset value.
Bits 3:2  **PRTCFG[1:0]: Protocol configuration**

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

- **00**: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
- **01**: SPDIF protocol
- **10**: AC'97 protocol
- **11**: Reserved

**Bits 1:0  **MODE[1:0]: SAIx audio block mode**

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

- **00**: Master transmitter
- **01**: Master receiver
- **10**: Slave transmitter
- **11**: Slave receiver

*Note*: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

### 56.6.3  **SAI configuration register 2 (SAI_ACR2)**

Address offset: 0x08

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

*Bits 31:16  **Reserved, must be kept at reset value.**

*Bits 15:14  **COMP[1:0]: Companding mode**

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on CPL bit.

- The data expansion or data compression are determined by the state of bit MODE[0].
- The data compression is applied if the audio block is configured as a transmitter.
- The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to Section: Companding mode for more details.

- **00**: No companding algorithm
- **01**: Reserved.
- **10**: µ-Law algorithm
- **11**: A-Law algorithm

*Note*: Companding mode is applicable only when Free protocol mode is selected.
Bit 13 **CPL:** Complement bit.
   This bit is set and cleared by software.
   It defines the type of complement to be used for companding mode
   0: 1’s complement representation.
   1: 2’s complement representation.
   
   **Note:** This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]:** Mute counter.
   These bits are set and cleared by software. They are used only in reception mode.
   The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.
   Refer to Section : Mute mode for more details.

Bit 6 **MUTEVAL:** Mute value.
   This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.
   If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.
   if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.
   Refer to Section : Mute mode for more details.
   0: Bit value 0 is sent during the mute mode.
   1: Last values are sent during the mute mode.
   
   **Note:** This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.
   This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.
   Refer to Section : Mute mode for more details.
   0: No mute mode.
   1: Mute mode enabled.
   
   **Note:** This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS:** Tristate management on data line.
   This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.
   Refer to Section : Output data line management on an inactive slot for more details.
   0: SD output line is still driven by the SAI when a slot is inactive.
   1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.
   This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.
   0: No FIFO flush.
   1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.
Bits 2:0 **FTH[2:0]**: FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

### 56.6.4 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C
Reset value: 0x0000 0007

**Note:** *This register has no meaning in AC’97 and SPDIF audio protocol.*

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

Bits 31:19 Reserved, must be kept at reset value.

**Bit 18** **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

**Bit 17** **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

**Bit 16** **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g. this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

**Bit 15** Reserved, must be kept at reset value.
Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock
(SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame.
These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number
of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio
block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot
0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be
aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not
used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.
These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration. They
must be configured when the audio block is disabled.

### 56.6.5 SAI slot register (SAI_ASLOTR)

**Address offset:** 0x10

**Reset value:** 0x0000 0000

**Note:** *This register has no meaning in AC’97 and SPDIF audio protocol.*

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:16 SLOTE[15:0]: Slot enable.**
These bits are set and cleared by software.
Each SLOTE bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

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

**Bits 11:8 NBSLOT[3:0]: Number of slots in an audio frame.**
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the
number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.
### Bits 7.6 SLOTSZ[1:0]: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

### Bits 4.0 FBOFF[4:0]: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

#### 56.6.6 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

**Bit 6 LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

**Bit 5 AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.
Bit 4 CNR DyIE: Codec not ready interrupt enable (AC’97).
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the
Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR
register is set and an interrupt is generated.
   This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio
block is operates as a receiver.

Bit 3 FREQIE: FIFO request interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
   Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be
configured before setting FREQIE to avoid a parasitic interrupt in receiver mode.

Bit 2 WCKCFGIE: Wrong clock configuration interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and
NODIV = 0.
   It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
   Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 MUTEDETIE: Mute detection interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
   This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 OVRUDR: Overrun/underrun interrupt enable.
   This bit is set and cleared by software.
   0: Interrupt is disabled
   1: Interrupt is enabled
   When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

56.6.7 SAI status register (SAI_ASR)

Address offset: 0x18
Reset value: 0x0000 0008
Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.

This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
- 000: FIFO empty (transmitter and receiver modes)
- 001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
- 010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
- 011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
- 100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
- 101: FIFO full (transmitter and receiver modes)

Others: Reserved

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

Bit 6 **LFSDET**: Late frame synchronization detection.

This bit is read only.
- 0: No error.
- 1: Frame synchronization signal is not present at the right time.

This flag can be set only if the audio block is configured in slave mode.
- It is not used in AC’97 or SPDIF mode.
- It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
- This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register.

Bit 5 **AFSDET**: Anticipated frame synchronization detection.

This bit is read only.
- 0: No error.
- 1: Frame synchronization signal is detected earlier than expected.

This flag can be set only if the audio block is configured in slave mode.
- It is not used in AC’97 or SPDIF mode.
- It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
- This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.

This bit is read only.
- 0: External AC’97 Codec is ready
- 1: External AC’97 Codec is not ready

This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
- It can generate an interrupt if CNRDYTE bit is set in SAI_xIM register.
- This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.

This bit is read only.
- 0: No FIFO request.
- 1: FIFO request to read or to write the SAI_xDR.

The request depends on the audio block configuration:
- If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
- If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.

This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.
**56.6.8 SAI clear flag register (SAI_ACLRFR)**

Address offset: 0x1C

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

**Bit 6 CLFSDET**: Clear late frame synchronization detection flag.

This bit is write only.

- Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
- This bit is not used in AC97 or SPDIF mode.
- Reading this bit always returns the value 0.
56.6.9 **SAI data register (SAI_ADR)**

Address offset: 0x20

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATA[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| 15:0 | DATA[15:0] |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

**Bits 31:0 DATA[31:0]: Data**

A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.
56.6.10 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24
Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN**: Master clock generation enable
- 0: The master clock is not generated
- 1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR**: Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
- 0: Master clock frequency = FFS x 256
- 1: Master clock frequency = FFS x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in Section 56.4.8: SAI clock generator.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV**: No divider
This bit is set and cleared by software.
- 0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
- 1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
- 0: DMA disabled
- 1: DMA enabled

Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.
Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
*Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.*

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

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
*Note: This bit has to be set before enabling the audio block and after the audio block configuration.*

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2.
When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to Section: Mono/stereo mode for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]**: Synchronization enable
These bits are set and cleared by software. They must be configured when the audio subblock is disabled.
00: audio subblock in asynchronous mode.
01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode
10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.
11: Reserved
*Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.*

Bit 9 **CKSTR**: Clock strobing edge
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.
0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.
1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST**: Least significant bit first
This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC’97 audio protocol since AC’97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.
0: Data are transferred with MSB first
1: Data are transferred with LSB first
Bits 7:5 **DS[2:0]: Data size**

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

- 000: Reserved
- 001: Reserved
- 010: 8 bits
- 011: 10 bits
- 100: 16 bits
- 101: 20 bits
- 110: 24 bits
- 111: 32 bits

**Bit 4** Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]: Protocol configuration**

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

- 00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.
- 01: SPDIF protocol
- 10: AC'97 protocol
- 11: Reserved

**Bits 1:0** **MODE[1:0]: SAIX audio block mode**

These bits are set and cleared by software. They must be configured when SAIX audio block is disabled.

- 00: Master transmitter
- 01: Master receiver
- 10: Slave transmitter
- 11: Slave receiver

**Note:** *When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.*

### 56.6.11 SAI configuration register 2 (SAI_BCR2)

**Address offset:** 0x28

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:16** Reserved, must be kept at reset value.
Bits 15:14 **COMP[1:0]:** Companding mode.

These bits are set and cleared by software. The μ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on CPL bit.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to **Section : Companding mode** for more details.

00: No companding algorithm
01: Reserved.
10: μ-Law algorithm
11: A-Law algorithm

*Note: Companding mode is applicable only when Free protocol mode is selected.*

Bit 13 **CPL:** Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.
1: 2’s complement representation.

*Note: This bit has effect only when the companding mode is μ-Law algorithm or A-Law algorithm.*

Bits 12:7 **MUTECNT[5:0]:** Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to **Section : Mute mode** for more details.

Bit 6 **MUTEVAL:** Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

If the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to **Section : Mute mode** for more details.

0: Bit value 0 is sent during the mute mode.
1: Last values are sent during the mute mode.

*Note: This bit is meaningless and must not be used for SPDIF audio blocks.*

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to **Section : Mute mode** for more details.

0: No mute mode.
1: Mute mode enabled.

*Note: This bit is meaningless and must not be used for SPDIF audio blocks.*
Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to Section: Output data line management on an inactive slot for more details.

0: SD output line is still driven by the SAI when a slot is inactive.
1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH**: FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.
1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled.

Bits 2:0 **FTH[2:0]**: FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

### 56.6.12 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

*Note:* This register has no meaning in AC’97 and SPDIF audio protocol

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>FSALL[6:0]</td>
<td>FRL[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.

This bit is set and cleared by software. It is meaningless and is not used in AC’97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.

0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.
Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC’97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOT register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC’97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.
These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.
The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOT register (NBSLOT[3:0] = 0000).
In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256. These bits are meaningless and are not used in AC’97 or SPDIF audio block configuration.

### 56.6.13 **SAI slot register (SAI_BSLOTR)**
Address offset: 0x30
Reset value: 0x0000 0000

*Note: This register has no meaning in AC’97 and SPDIF audio protocol.*
56.6.14 **SAI interrupt mask register (SAI_BIM)**

Address offset: 0x34

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:16 SLOTEN[15:0]: Slot enable.**

These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

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

**Bits 11:8** NBSLOT[3:0]: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots + 1 in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.

**Bits 7:6** SLOTSZ[1:0]: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to Output data line management on an inactive slot for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

**Bit 5** Reserved, must be kept at reset value.

**Bits 4:0** FBOFF[4:0]: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC’97 or SPDIF mode.
Bits 31:7  Reserved, must be kept at reset value.

- **Bit 6  LFSDETIE**: Late frame synchronization detection interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
  - This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

- **Bit 5  AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
  - This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

- **Bit 4  CNRDYIE**: Codec not ready interrupt enable (AC’97).
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
  - This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

- **Bit 3  FREQIE**: FIFO request interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
  - Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode.

- **Bit 2  WCKCFGIE**: Wrong clock configuration interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
  - It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
  - Note: This bit is used only in Free protocol mode and is meaningless in other modes.

- **Bit 1  MUTEDETIE**: Mute detection interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
  - This bit has a meaning only if the audio block is configured in receiver mode.

- **Bit 0  OVRUDRIE**: Overrun/underrun interrupt enable.
  - This bit is set and cleared by software.
  - 0: Interrupt is disabled
  - 1: Interrupt is enabled
  - When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.
## 56.6.15 SAI status register (SAI_BSR)

Address offset: 0x38  
Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>Bit 31:19 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 18:16 <strong>FLVL[2:0]</strong>: FIFO level threshold.</td>
</tr>
<tr>
<td>This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).</td>
</tr>
<tr>
<td>000: FIFO empty (transmitter and receiver modes)</td>
</tr>
<tr>
<td>001: FIFO (\leq \frac{1}{4}) but not empty (transmitter mode), FIFO (\leq \frac{1}{4}) but not empty (receiver mode)</td>
</tr>
<tr>
<td>010: (\frac{1}{4} &lt; \text{FIFO} \leq \frac{1}{2}) (transmitter mode), (\frac{1}{4} &lt; \text{FIFO} &lt; \frac{1}{2}) (receiver mode)</td>
</tr>
<tr>
<td>011: (\frac{1}{2} &lt; \text{FIFO} &lt; \frac{3}{4}) (transmitter mode), (\frac{1}{2} &lt; \text{FIFO} &lt; \frac{3}{4}) (receiver mode)</td>
</tr>
<tr>
<td>100: (\frac{3}{4} &lt; \text{FIFO} &lt; \text{full (transmitter mode), } \frac{3}{4} &lt; \text{FIFO but not full (receiver mode)}</td>
</tr>
<tr>
<td>101: FIFO full (transmitter and receiver modes)</td>
</tr>
<tr>
<td>Others: Reserved</td>
</tr>
</tbody>
</table>

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

| Bit 6 **LFSDET**: Late frame synchronization detection. |
| This bit is read only. |
| 0: No error. |
| 1: Frame synchronization signal is not present at the right time. |
| This flag can be set only if the audio block is configured in slave mode. |
| It is not used in AC’97 or SPDIF mode. |
| It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register. |
| This flag is cleared when the software sets CLFSDET in SAI_xCLRFR register |

| Bit 5 **AFSDET**: Anticipated frame synchronization detection. |
| This bit is read only. |
| 0: No error. |
| 1: Frame synchronization signal is detected earlier than expected. |
| This flag can be set only if the audio block is configured in slave mode. |
| It is not used in AC’97 or SPDIF mode. |
| It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register. |
| This flag is cleared when the software sets CAFSDET in SAI_xCLRFR register |
Bit 4 **CNRYD**: Codec not ready.
   This bit is read only.
   0: External AC’97 Codec is ready
   1: External AC’97 Codec is not ready
   This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
   It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
   This bit is read only.
   0: No FIFO request.
   1: FIFO request to read or to write the SAI_xDR.
   The request depends on the audio block configuration:
   – If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
   – If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
   This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
   This bit is read only.
   0: Clock configuration is correct
   1: Clock configuration does not respect the rule concerning the frame length specification defined in Section 56.4.6: Frame synchronization (configuration of FRL[7:0] bit in the SAI_xFRCR register)
   This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
   It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
   This bit is read only.
   0: No MUTE detection on the SD input line
   1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
   This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
   It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
   This bit is read only.
   0: No overrun/underrun error.
   1: Overrun/underrun error detection.
   The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
   It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
   This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.
56.6.16 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 CLFSDET: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 5 CAFSDET: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 CCNRDY: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 CWCKCFG: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 CMUTEDET: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 COVRUDR: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.
### 56.6.17 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**DATA[31:16]**

Bits 31:0 **DATA[31:0]**: Data

A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

### 56.6.18 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**DATA[15:0]**

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

- **Bit 9 CKEN2**: Clock enable of bitstream clock number 2
  - This bit is set and cleared by software.
  - 0: SAI_CK2 clock disabled
  - 1: SAI_CK2 clock enabled
  - **Note**: It is not recommended to configure this bit when PDMEN = 1.
    - SAI_CK2 might not be available for all SAI instances. Refer to Section 56.3: SAI implementation for details.

- **Bit 8 CKEN1**: Clock enable of bitstream clock number 1
  - This bit is set and cleared by software.
  - 0: SAI_CK1 clock disabled
  - 1: SAI_CK1 clock enabled
  - **Note**: It is not recommended to configure this bit when PDMEN = 1.
    - SAI_CK1 might not be available for all SAI instances. Refer to Section 56.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.
Bits 5:4 **MICNR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones
01: Configuration with 4 microphones
10: Configuration with 6 microphones
11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.*

The complete set of data lines might not be available for all SAI instances. Refer to Section 56.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI is already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled
1: PDM interface enabled

### 56.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bitfield is set and cleared by software.

000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

*Note: This bitfield can be used only if D4 line is available. Refer to Section 56.3: SAI implementation to check if it is available.*

Bit 27 Reserved, must be kept at reset value.
Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4
This bitfield is set and cleared by software.

- 000: No delay
- 001: Delay of 1 $T_{SAI\_CK}$ period
- 010: Delay of 2 $T_{SAI\_CK}$ periods
...
- 111: Delay of 7 $T_{SAI\_CK}$ periods

This bitfield can be changed on-the-fly.
*Note: This bitfield can be used only if D4 line is available. Refer to Section 56.3: SAI implementation to check if it is available.*

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.

- 000: No delay
- 001: Delay of 1 $T_{SAI\_CK}$ period
- 010: Delay of 2 $T_{SAI\_CK}$ periods
...
- 111: Delay of 7 $T_{SAI\_CK}$ periods

This bitfield can be changed on-the-fly.
*Note: This bitfield can be used only if D3 line is available. Refer to Section 56.3: SAI implementation to check if it is available.*

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.

- 000: No delay
- 001: Delay of 1 $T_{SAI\_CK}$ period
- 010: Delay of 2 $T_{SAI\_CK}$ periods
...
- 111: Delay of 7 $T_{SAI\_CK}$ periods

This bitfield can be changed on-the-fly.
*Note: This bitfield can be used only if D3 line is available. Refer to Section 56.3: SAI implementation to check if it is available.*

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.

- 000: No delay
- 001: Delay of 1 $T_{SAI\_CK}$ period
- 010: Delay of 2 $T_{SAI\_CK}$ periods
...
- 111: Delay of 7 $T_{SAI\_CK}$ periods

This bitfield can be changed on-the-fly.
*Note: This bitfield can be used only if D2 line is available. Refer to Section 56.3: SAI implementation to check if it is available.*

Bit 11 Reserved, must be kept at reset value.
Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2

This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI\_CK} period
010: Delay of 2 T_{SAI\_CK} periods
...
111: Delay of 7 T_{SAI\_CK} periods

This bitfield can be changed on-the-fly.
*Note:* This bitfield can be used only if D2 line is available. Refer to Section 56.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1

This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI\_CK} period
010: Delay of 2 T_{SAI\_CK} periods
...
111: Delay of 7 T_{SAI\_CK} periods

This bitfield can be changed on-the-fly.
*Note:* This bitfield can be used only if D1 line is available. Refer to Section 56.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1

This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI\_CK} period
010: Delay of 2 T_{SAI\_CK} periods
...
111: Delay of 7 T_{SAI\_CK} periods

This bitfield can be changed on-the-fly.
*Note:* This bitfield can be used only if D1 line is available. Refer to Section 56.3: SAI implementation to check if it is available.

### 56.6.20 SAI register map

#### Table 598. SAI register map and reset values

| Offset   | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x0000   | SAI_GCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
### Table 598. SAI register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x04 or 0x24</td>
<td>SAI_xCR1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>0x08 or 0x28</td>
<td>SAI_xCR2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0C or 0x2C</td>
<td>SAI_xFRCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0x10 or 0x30</td>
<td>SAI_xSLOTR</td>
<td>SLOTEN[15:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x14 or 0x34</td>
<td>SAI_xIM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x18 or 0x38</td>
<td>SAI_xSR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x1C or 0x3C</td>
<td>SAI_xCLFRFR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x20 or 0x40</td>
<td>SAI_xDR</td>
<td>DATA[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x44</td>
<td>SAI_PDMCR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x48</td>
<td>SAI_PDMRDL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

Refer to Section 2.3 for the register boundary addresses.
57 SPDIF receiver interface (SPDIFRX)

57.1 SPDIFRX interface introduction
The SPDIFRX interface handles S/PDIF audio protocol.

57.2 SPDIFRX main features
- Up to 4 inputs available
- Automatic symbol rate detection
- Maximum symbol rate: 12.288 MHz
- Stereo stream from 8 to 192 kHz\(^{(a)}\) supported
- Supports audio IEC-60958 and IEC-61937, consumer applications
- Parity bit management
- Communication using DMA for audio samples
- Communication using DMA for control and user channel information
- Interrupt capabilities

57.3 SPDIFRX functional description
The SPDIFRX peripheral, is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multi-channel surround sound, such as those defined by Dolby or DTS.

The receiver provides all the necessary features to detect the symbol rate, and decode the incoming data. It is possible to use a dedicated path for the user and channel information in order to ease the interface handling. Figure 830 shows a simplified block diagram.

The SPDIFRX_DC block is responsible of the decoding of the S/PDIF stream received from SPDIFRX_IN[3:0] inputs. This block re-sample the incoming signal, decode the manchester stream, recognize frames, sub-frames and blocks elements. It delivers to the REG_IF part, decoded data, and associated status flags.

This peripheral can be fully controlled via the APB bus, and can handle two DMA channels:
- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user information

Interrupt services are also available either as an alternative function to the DMA, or for signaling error or key status of the peripheral.

The SPDIFRX also offers a signal named spdifrx_frame_sync, which toggles every time that a sub-frame’s preamble is detected. So the duty cycle is 50%, and the frequency equal to the frame rate.

This signal can be connected to timer events, in order to compute frequency drift.

\(\text{a. Check the RCC capabilities in order to verify which sampling rates can be supported.}\)
In addition the SPDIFRX also provides a signal named `spdifrx_symb_ck` toggling at the symbol rate.

**Figure 830. SPDIFRX block diagram**

1. 'n' is fixed to 3, and 'x' is set to 0.

### 57.3.1 SPDIFRX pins and internal signals

*Table 599* lists the SPDIFRX internal input/output signals, *Table 600* the SPDIFRX pins (alternate functions).

**Table 599. SPDIFRX internal input/output signals**

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spdifrx_ker_ck</code></td>
<td>Digital input</td>
<td>SPDIFRX kernel clock</td>
</tr>
<tr>
<td><code>spdifrx_pclk</code></td>
<td>Digital input</td>
<td>SPDIFRX register interface clock</td>
</tr>
<tr>
<td><code>spdifrx_it</code></td>
<td>Digital output</td>
<td>SPDIFRX global interrupt</td>
</tr>
<tr>
<td><code>spdifrx_dat_dma</code></td>
<td>Digital input/output</td>
<td>SPDIFRX DMA request (and acknowledge) for data transfer</td>
</tr>
<tr>
<td><code>spdifrx_ctrl_dma</code></td>
<td>Digital input/output</td>
<td>SPDIFRX DMA request (and acknowledge) for channel status and user information transfer</td>
</tr>
<tr>
<td><code>spdifrx_frame_sync</code></td>
<td>Digital output</td>
<td>SPDIFRX frame rate synchronization signal</td>
</tr>
<tr>
<td><code>spdifrx_symb_ck</code></td>
<td>Digital output</td>
<td>SPDIFRX channel symbol clock</td>
</tr>
</tbody>
</table>

**Table 600. SPDIFRX pins**

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Pin type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDIFRX_IN0</td>
<td>Digital input</td>
<td>Input 0 for S/PDIF signal</td>
</tr>
<tr>
<td>SPDIFRX_IN1</td>
<td>Digital input</td>
<td>Input 1 for S/PDIF signal</td>
</tr>
</tbody>
</table>
57.3.2 S/PDIF protocol (IEC-60958)

S/PDIF block

A S/PDIF frame is composed of two sub-frames (see Figure 831). Each sub-frame contains 32 bits (or time slots):

- Bits 0 to 3 carry one of the synchronization preambles
- Bits 4 to 27 carry the audio sample word in linear 2's complement representation. The most significant bit (MSB) is carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8.
- Bit 28 (validity bit “V”) indicates if the data is valid (for converting it to analog for example)
- Bit 29 (user data bit “U”) carries the user data information like the number of tracks of a Compact Disk.
- Bit 30 (channel status bit “C”) carries the channel status information like sample rate and protection against copy.
- Bit 31 (parity bit “P”) carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity).

**Figure 831. S/PDIF sub-frame format**

For linear coded audio applications, the first sub-frame (left or “A” channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble “M”. However, the preamble changes to preamble “B” once every 192 frames to identify the start of the block structure used to organize the channel status and user information. The second sub-frame (right or “B” channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble “W”.

A S/PDIF block contains 192 pairs of sub-frames of 32 bits.

Table 600. SPDIFRX pins (continued)

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Pin type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPDIFRX_IN2</td>
<td>Digital input</td>
<td>Input 2 for S/PDIF signal</td>
</tr>
<tr>
<td>SPDIFRX_IN3</td>
<td>Digital input</td>
<td>Input 3 for S/PDIF signal</td>
</tr>
</tbody>
</table>
Synchronization preambles

The preambles patterns are inverted or not according to the previous half-bit value. This previous half-bit value is the level of the line before enabling a transfer for the first “B” preamble of the first frame. For the others preambles, this previous half-bit value is the second half-bit of the parity bit of the previous sub-frame. The preambles patterns B, M and W are described in the Figure 833.

Figure 833. S/PDIF Preambles

Coding of information bits

In order to minimize the DC component value on the transmission line, and to facilitate clock recovery from the data stream, bits 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is
logical 0. However, it is different if the bit is logical 1. These states are named “UI” (unit interval) in the IEC-60958 specification.

The 24 data bits are transferred LSB first.

Figure 834. Channel coding example

57.3.3 SPDIFRX decoder (SPDIFRX_DC)

Main principle

The technique used by the SPDIFRX in order to decode the S/PDIF stream is based on the measurement of the time interval between two consecutive edges. Three kinds of time intervals may be found into an S/PDIF stream:

- The long time interval, having a duration of 3 x UI, noted TL. It appears only during preambles.
- The medium time interval, having a duration of 2 x UI, noted TM. It appears both in some preambles or into the information field.
- The short time interval, having a duration of 1 x UI, noted TS. It appears both in some preambles or into the information field.

The SPDIFRX_DC block is responsible of the decoding of the received S/PDIF stream. It takes care of the following functions:

- Resampling and filtering of the incoming signal
- Estimation of the time-intervals
- Estimation of the symbol rate and synchronization
- Decoding of the serial data, and check of integrity
- Detection of the block, and sub-frame preambles
- Continuous tracking of the symbol rate
Figure 835 gives a detailed view of the SPDIFRX decoder.

Figure 835. SPDIFRX decoder

1. 'n' is fixed to 3, and 'x' is set to 0.

Noise filtering and rising/falling edge detection

The S/PDIF signal received on the selected SPDIFRX_IN is re-sampled using the spdifrx_ker_ck clock (acquisition clock). A simple filtering is applied in order cancel spurs. This is performed by the stage detecting the edge transitions. The edge transitions are detected as follow:

- A rising edge is detected when the sequence 0 followed by two 1 is sampled.
- A falling edge is detected when the sequence 1 followed by two 0 is sampled.
- After a rising edge, a falling edge sequence is expected.
- After a falling edge, a rising edge sequence is expected.

Figure 836. Noise filtering and edge detection

Longest and shortest transition detector

The longest and shortest transition detector block detects the maximum (MAX_CNT) and minimum (MIN_CNT) duration between two transitions. The TRCNT counter is used to measure the time interval duration. It is clocked by the spdifrx_ker_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again.
The maximum duration is normally found during the preamble period. This maximum duration is sent out as MAX_CNT. The minimum duration is sent out as MIN_CNT.

The search of the longest and shortest transition is stopped when the transition timer expires. The transition timer is like a watchdog timer that generates a trigger after 70 transitions of the incoming signal. Note that counting 70 transitions insures a delay a bit longer than a sub-frame.

Note that when the TRCNT overflows due to a too long time interval between two pulses, the SPDIFRX is stopped and the flag TERR of SPDIFRX_SR register is set to 1.

### Transition coder and preamble detector

The transition coder and preamble detector block receives the MAX_CNT and MIN_CNT. It also receives the current transition width from the TRCNT counter (see Figure 835). This block encodes the current transition width by comparing the current transition width with two different thresholds, names TH\textsubscript{HI} and TH\textsubscript{LO}.

- If the current transition width is less than (TH\textsubscript{LO} - 1), then the data received is half part of data bit ‘1’, and is coded as TS.
- If the current transition width is greater than (TH\textsubscript{LO} - 1), and less than TH\textsubscript{HI}, then the data received is data bit ‘0’, and is coded as TM.
- If the current transition width is greater than TH\textsubscript{HI}, then the data received is the long pulse of preambles, and is coded as TL.
- Else an error code is generated (FERR flag is set).

The thresholds TH\textsubscript{HI} and TH\textsubscript{LO} are elaborated using two different methods.

If the peripheral is doing its initial synchronization (‘coarse synchronization’), then the thresholds are computed as follow:

- TH\textsubscript{LO} = \text{MAX\_CNT} / 2.
- TH\textsubscript{HI} = \text{MIN\_CNT} + \text{MAX\_CNT} / 2.

Once the ‘coarse synchronization’ is completed, then the SPDIFRX uses a more accurate reference in order to elaborate the thresholds. The SPDIFRX measures the length of 24 symbols (WIDTH24) for defining TH\textsubscript{LO} and the length of 40 symbols (WIDTH40) for TH\textsubscript{HI}. TH\textsubscript{HI} and TH\textsubscript{LO} are computed as follow:

- TH\textsubscript{LO} = (WIDTH24) / 32
- TH\textsubscript{HI} = (WIDTH40) / 32

This second synchronization phase is called the ‘fine synchronization’. Refer to Figure 839 for additional information.
As shown in the figure hereafter, TH_{LO} is ideally equal to 1.5 UI, and to TH_{HI} 2.5 UI.

**Figure 837. Thresholds**

The preamble detector checks four consecutive transitions of a specific sequence to determine if they form the part of preamble. Let us say TRANS0, TRANS1, TRANS2 and TRANS3 represent four consecutive transitions encoded as mentioned above. *Table 601* shows the values of these four transitions to form a preamble. Absence of this pattern indicates that these transitions form part of the data in the sub frame and bi-phase decoder decode them.

### Table 601. Transition sequence for preamble

<table>
<thead>
<tr>
<th>Preamble type</th>
<th>Biphasic data pattern</th>
<th>TRANS3</th>
<th>TRANS2</th>
<th>TRANS1</th>
<th>TRANS0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble B</td>
<td>11101000</td>
<td>TL</td>
<td>TS</td>
<td>TS</td>
<td>TL</td>
</tr>
<tr>
<td>Preamble M</td>
<td>11100100</td>
<td>TL</td>
<td>TL</td>
<td>TS</td>
<td>TS</td>
</tr>
<tr>
<td>Preamble W</td>
<td>11100100</td>
<td>TL</td>
<td>TM</td>
<td>TS</td>
<td>TM</td>
</tr>
</tbody>
</table>

**Bi-phase decoder**

The Bi-phase decoder decodes the input bi-phase marked data stream using the transition information provided by the *transition coder and preamble detector* block. It first waits for the preamble detection information. After the preamble detection, it decodes the following transition information:

- If the incoming transition information is TM then it is decoded as a ‘0’.
- Two consecutive TS are decoded as a ‘1’.
- Any other transition sequence generates an error signal (FERR set to 1).

After decoding 28 data bits this way, this module looks for the following preamble data. If the new preamble is not what is expected, then this block generates an error signal (FERR set to 1). Refer to *Section 57.3.9: Reception errors*, for additional information on error flags.

**Data packing**

This block is responsible of the decoding of the IEC-60958 frames and blocks. It also handles the writing into the RX_BUF or into SPDIFRX_CSR register.
57.3.4 SPDIFRX tolerance to clock deviation

The SPDIFRX tolerance to clock deviation depends on the number of sample clock cycles in one bit slot. The fastest spdifrx_ker_ck is, the more robust the reception is. The ratio between spdifrx_ker_ck frequency and the symbol rate must be at least 11.

Two kinds of phenomenon (at least) can degrade the reception quality:

- The cycle-to-cycle jitter which reflects the difference of transition length between two consecutive transitions.
- The long term jitter which reflects a cumulative effect of the cycle-to-cycle jitter. It can be seen as a low-frequency symbol modulation.

57.3.5 SPDIFRX synchronization

The synchronization phase starts when setting SPDIFRXEN to 01 or 11. Figure 838 shows the synchronization process.

If the bit WFA of SPDIFRX_CR register is set to 1, then the peripheral must first detect activity on the selected SPDIFRX_IN line before starting the synchronization process. The activity detection is performed by detecting four transitions on the selected SPDIFRX_IN. The peripheral remains in this state until transitions are not detected. This function can be particularly helpful because the SPDIFRX switches in COARSE SYNC mode only if activity is present on the selected SPDIFRX_IN input, avoiding synchronization errors. See Section 57.4: Programming procedures for additional information.

The user can still set the SPDIFRX into STATE_IDLE by setting SPDIFRXEN to 0. If the WFA is set to 0, the peripheral starts the coarse synchronization without checking activity.

The next step consists on doing a first estimate of the thresholds (COARSE SYNC), in order to perform the fine synchronization (FINE SYNC). Due to disturbances of the SPDIFRX line, it can happen that the process is not executed first time right. For this purpose, the user can program the number of allowed re-tries (NBTR) before setting SERR error flag.

When the SPDIFRX is able to measure properly the duration of 24 and 40 consecutive symbols then the FINE SYNC is completed, the threshold values are updated, and the flag SYNCD is set to 1. Refer to Section : Transition coder and preamble detector for additional information.

Two kinds of errors are detected:

- An overflow of the TRCNT, which generally means that there is no valid S/PDIF stream in the input line. This overflow is indicated by TERR flag.
- The number of retries reached the programmed value. This means that strong jitter is present on the S/PDIF signal. This error is indicated by SERR flag.

When the first FINE SYNC is completed, the reception of channel status (C) and user data (U) starts when the next “B” preamble is detected (see Figure 842). Then the user can read IEC-60958 C and U bits through SPDIFRX_CSR register. According to this information the user can then select the proper settings for DRFMT and RXSTEO. For example if the user detects that the current audio stream transports encoded data, then he can put RXSTEO to 0, and DRFMT to 10 prior to start data reception. Note that DRFMT and RXSTEO cannot be modified when SPDIFRXEN = 11. Writes to these fields are ignored if SPDIFRXEN is already 11, though these field can be changed with the same write instruction that causes SPDIFRXEN to become 11.

Then the SPDIFRX waits for SPDIFRXEN = 11 and the “B” preamble before starting saving audio samples.
Refer to Frame structure and synchronization error for additional information concerning TRCNT overflow.

The FINE SYNC process is re-triggered every frame in order to update thresholds as shown in Figure 839 in order to continuously track S/PDIF synchronization.
57.3.6 SPDIFRX handling

The software can control the state of the SPDIFRX through SPDIFRXEN field. The SPDIFRX can be into one of the following states:

- **STATE_IDLE:**
  The peripheral is disabled, the spdifrx_ker_ck domain is reset. The spdifrx_pclk domain is functional.

- **STATE_SYNC:**
  The peripheral is synchronized to the stream, thresholds are updated regularly, user and channel status can be read via interrupt of DMA. The audio samples are not provided to receive buffer.

- **STATE_RCV:**
  The peripheral is synchronized to the stream, thresholds are updated regularly, user, channel status and audio samples can be read via interrupt or DMA channels. When SPDIFRXEN goes to 11, the SPDIFRX waits for “B” preamble before starting saving audio samples.

- **STOP_STATE:**
  The peripheral is no longer synchronized, the reception of the user, channel status and audio samples are stopped. It is expected that the software re-starts the SPDIFRX.

*Figure 840* shows the possible states of the SPDIFRX, and how to transition from one state to the other. The bits under software control are followed by the mention “(SW)”, the bits under SPDIFRX control are followed by the mention “(HW)”.

---

**Figure 839. Synchronization process scheduling**

![Figure 839: Synchronization process scheduling](image-url)
When SPDIFRX is in STATE_IDLE:

- The software can transition to STATE_SYNC by setting SPDIFRXEN to 01 or 11.

When SPDIFRX is in STATE_SYNC:

- If the synchronization fails or if the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- When the synchronization phase is completed, if SPDIFRXEN = 01 the peripheral remains in this state.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.
- The SPDIFRX goes to STATE_RCV if SPDIFRXEN = 11 and if the SYNCD = 1.

When SPDIFRX is in STATE_RCV:

- If the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.

When SPDIFRX is in STATE_STOP:

- The SPDIFRX stops reception and synchronization, and waits for the software to set the bit SPDIFRXEN to 0, in order to clear the error flags.
When SPDIFRXEN is set to 0, the SPDIFRX is disabled, meaning that all the state machines are reset, and RX_BUF is flushed. Note as well that flags FERR, SERR and TERR are reset.

57.3.7 Data reception management

The SPDIFRX offers a double buffer for the audio sample reception. A 32-bit buffer located into the spdifrx_ker_ck clock domain (RX_BUF), and the SPDIFRX_FMTx_DR register. The valid data contained into the RX_BUF are immediately transferred into SPDIFRX_FMTx_DR if SPDIFRX_FMTx_DR is empty.

The valid data contained into the RX_BUF are transferred into SPDIFRX_FMTx_DR when the two following conditions are reached:

- The transition between the parity bit (P) and the next preamble is detected (this indicated that the word is completely received).
- The SPDIFRX_FMTx_DR is empty.

Having a 2-word buffer gives more flexibility for the latency constraint.

The maximum latency allowed is $T_{SAMPLE} - 2T_{PCLK} - 2T_{spdifrx_ker_ck}$

Where $T_{SAMPLE}$ is the audio sampling rate of the received stereo audio samples, $T_{PCLK}$ is the period of spdifrx_pclk clock, and $T_{spdifrx_ker_ck}$ is the period of spdifrx_ker_ck clock.

The SPDIFRX offers the possibility to use either DMA (spdifrx_dat_dma and spdifrx_ctrl_dma) or interrupts for transferring the audio samples into the memory. The recommended option is DMA, refer to Section 57.3.12: DMA interface for additional information.

The SPDIFRX offers several way on handling the received data. The user can either have a separate flow for control information and audio samples, or get them all together.

For each sub-frame, the data reception register SPDIFRX_FMTx_DR contains the 24 data bits, and optionally the V, U, C, PE status bits, and the PT (see Mixing data and control flow).

Note that PE bit stands for parity error bit, and is set to 1 when a parity error is detected in the decoded sub-frame.

The PT field carries the preamble type (B, M or W).

V, U and C are a direct copy of the value received from the S/PDIF interface.

The bit DRFMT allows the selection between 3 audio formats as shown in Figure 841.

This document describes 3 data registers: SPDIFRX_FMTx[2:0] (x = 2 to 0), but in reality there is only one physical data register, having 3 possible formats:

- When DRFMT = 0, the format of the data register is the one described by SPDIFRX_FMT0_DR
- When DRFMT = 1, the format of the data register is the one described by SPDIFRX_FMT1_DR
- When DRFMT = 2, the format of the data register is the one described by SPDIFRX_FMT2_DR
Setting DRFMT to 00 or 01, offers the possibility to have the data either right or left aligned into the SPDIFRX_FMTx_DR register. The status information can be enabled or forced to zero according to the way the software wants to handle them.

The format given by DRFMT = 10 is interesting in non-linear mode, as only 16 bits per sub-frame are used. By using this format, the data of two consecutive sub-frames are stored into SPDIFRX_FMTx_DR, dividing by two the amount of memory footprint. Note that when RXSTEO = 1, there is no misalignment risks (i.e. data from ChA are always stored into SPDIFRX_FMTx_DR[31:16]). If RXSTEO = 0, then there is a misalignment risk is case of overrun situation. In that case SPDIFRX_FMTx_DR[31:16] always contain the oldest value and SPDIFRX_FMTx_DR[15:0] the more recent value (see Figure 843).

In this format the status information cannot be mixed with data, but the user can still get them through SPDIFRX_CSR register, and use a dedicated DMA channel or interrupt to transfer them to memory (see Section 57.3.8: Dedicated control flow)

**Mixing data and control flow**

The user can choose to use this mode in order to get the full flexibility of the handling of the control flow. The user can select which field must be kept into the data register (SPDIFRX_FMTx_DR).

- When bit PMSK = 1, the parity error information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit VMSK = 1, the validity information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit CUMSK = 1, the channel status, and used data information are masked (set to 0), otherwise they are copied into SPDIFRX_FMTx_DR.
- When bit PTMSK = 1, the preamble type is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

57.3.8 Dedicated control flow

The SPDIFRX offers the possibility to catch both user data and channel status information via a dedicated DMA channel. This feature allows the SPDIFRX to acquire continuously the channel status and user information. The acquisition starts at the beginning of a IEC 60958 block. Two fields are available to control this path: CBDMAEN and SPDIFRXEN. When SPDIFRXEN is set to 01 or 0x11, the acquisition is started, after completion of the synchronization phase. When 8 channel status and 16 user data bits are received, they are packed and stored into SPDIFRX_CSR register. A DMA request is triggered if the bit CBDMAEN is set to 1 (see Figure 842).

If CS[0] corresponds to the first bit of a new block, the bit SOB is set to 1. Refer to Section 57.5.8: SPDIFRX channel status register (SPDIFRX_CSR). A bit is available (CHSEL) in order to select if the user wants to select channel status information (C) from the channel A or B.

![Figure 842. Channel/user data format](image)

Note: Once the first start of block is detected (B preamble), the SPDIFRX is checking the preamble type every 8 frames.

Note: Overrun error on SPDIFRX_FMTx_DR register does not affect this path.
57.3.9  Reception errors

Frame structure and synchronization error

The SPDIFRX, detects errors, when one of the following condition occurs:

- The FERR bit is set to 1 on the following conditions:
  - For each of the 28 information bits, if one symbol transition sequence is not correct: for example if short pulses are not grouped by pairs.
  - If preambles occur to an unexpected place, or an expected preamble is not received.
- The SERR bit is set when the synchronization fails, because the number of re-tries exceeded the programmed value.
- The TERR bit is set when the counter used to estimate the width between two transitions overflows (TRCNT).
  The overflow occurs when no transition is detected during 8192 periods of spdifrx_ker_ck clock. It represents at most a time interval of 11.6 frames.

When one of those flags goes to 1, the traffic on selected SPDIFRX_IN is then ignored, an interrupt is generated if the IFEIE bit of the SPDIFRX_CR register is set.

The normal procedure when one of those errors occur is:

- Set SPDIFRXEN to 0 in order to clear the error flags
- Set SPDIFRXEN to 01 or 11 in order to restart the SPDIFRX

Refer to Figure 840 for additional information.

Parity error

For each sub-frame, an even number of zeros and ones is expected inside the 28 information bits. If not, the parity error bit PERR is set in the SPDIFRX_SR register and an interrupt is generated if the parity interrupt enable PERRIE bit is set in the SPDIFRX_CR register. The reception of the incoming data is not paused, and the SPDIFRX continue to deliver data to SPDIFRX_FMTx_DR even if the interrupt is still pending.

The interrupt is acknowledged by clearing the PERR flag through PERRCF bit.

If the software wants to guarantee the coherency between the data read in the SPDIFRXFmtx_DR register and the value of the bit PERR, the bit PMSK must be set to 0.

Overrun error

If both SPDIFRX_FMTx_DR and RX_BUF are full, while the SPDIFRX_DC needs to write a new sample in RX_BUF, this new sample is dropped, and an overrun condition is triggered. The overrun error flag OVR is set in the SPDIFRX_SR register and an interrupt is generated if the OVRIE bit of the SPDIFRX_CR register is set.

If the RXSTEO bit is set to 0, then as soon as the RX_BUF is empty, the SPDIFRX stores the next incoming data, even if the OVR flag is still pending. The main purpose is to reduce as much as possible the amount of lost samples. Note that the behavior is similar independently of DRFMT value. See Figure 843.
If the RXSTEO bit is set to 1, it means that stereo data are transported, then the SPDIFRX has to avoid misalignment between left and right channels. So the peripheral has to drop a second sample even if there is room inside the RX_BUF in order to avoid misalignment. Then the incoming samples can be written normally into the RX_BUF even if the OVR flag is still pending. Refer to Figure 844.

The OVR flag is cleared by software, by setting the OVRCF bit to 1.
57.3.10 Clocking strategy

The SPDIFRX block needs two different clocks:

- The APB clock (spdifrx_pclk), which is used for the register interface,
- The spdifrx_ker_ck which is mainly used by the SPDIFRX_DC part. Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized (SYNC block on Figure 830).

In order to decode properly the incoming S/PDIF stream the SPDIFRX_DC must re-sample the received data with a clock at least 11 times higher than the maximum symbol rate, or 704 times higher than the audio sample rate. For example if the user expects to receive a symbol rate up to 12.288 MHz, the sample rate must be at least 135.2 MHz. The clock used by the SPDIFRX_DC is the spdifrx_ker_ck.

The frequency of the spdifrx_pclk must be at least equal to the symbol rate.

### Table 602. Minimum spdifrx_ker_ck frequency versus audio sampling rate

<table>
<thead>
<tr>
<th>Symbol rate</th>
<th>Minimum spdifrx_ker_ck frequency</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.072 MHz</td>
<td>33.8 MHz</td>
<td>For 48 kHz stream</td>
</tr>
<tr>
<td>6.144 MHz</td>
<td>67.6 MHz</td>
<td>For 96 kHz stream</td>
</tr>
<tr>
<td>12.288 MHz</td>
<td>135.2 MHz</td>
<td>For 192 kHz stream</td>
</tr>
</tbody>
</table>

1. Check the RCC capabilities in order to verify which sampling rates can be supported.

57.3.11 Symbol clock generation

The SPDIFRX block provides a symbol clock on signal named spdifrx_symb_ck, which can be used as the reference kernel clock for another audio device such as SAI or SPI/I2S. It can be used for SPDIFRX to I2S bridge function.
The symbol clock is built using the values of WIDTH24, WIDTH40 and the symbol boundaries.

- During the reception of the sub-frame sync preambles, the falling and rising edges of the symbol clock are built from the WIDTH24 and WIDTH40 values. Note that WIDTH24 and WIDTH40 are also used for the generation of the symbol clock, when the SPDIFRX is STATE_STOP or STATE_IDLE. See Table 603 for details.
- During the reception of the sub-frame payload, the SPDIFRX uses the symbol boundaries to generate the rising edge, the WIDTH24 and WIDTH40 values for the generation of the falling edge.

The duty cycle of the symbol clock is close to 50% during the reception of the sub-frame payload. However, the duty cycle can be altered when the SPDIFRX transitions from a symbol clock generated with WIDTH24 and WIDTH40 to a clock generated by the symbol clock boundaries or vice-versa.

The symbol clock has an important jitter mainly due to:

- The re-sampling of the S/PDIF signal with spdifrx_ker_ck clock
- The transition of the symbol clock generation mode

For that reason the application must consider the quality degradation if the symbol clock is used as the reference clock for A/D or D/A converters.

The generation of this symbol clock is controlled by the CKSEN bit. When CKSEN = ‘1’, the clock symbol is generated when the SPDIFRX completes successfully the first fine synchronization (SYNCD = 1), and when it receives correct data from the selected SPDIFRX input.

When the SPDIFRX goes to STATE_STOP, or STATE_IDLE, the symbol clock is gated if the bit CKSBKPEN = ‘0’. If the CKSBKPEN = ‘1’, then a backup symbol clock is still generated if the SPDIFRX is properly synchronized (i.e. valid values available for WIDTH24 and WIDTH40). Table 603 gives more details on the conditions controlling the generation of the symbol clock.

Table 603. Conditions of spdifrx_symb_ck generation

<table>
<thead>
<tr>
<th>SPDIFRX states and conditions</th>
<th>CKSEN</th>
<th>CKSBKPEN</th>
<th>spdifrx_symb_ck state</th>
</tr>
</thead>
<tbody>
<tr>
<td>Any state</td>
<td>0</td>
<td>X</td>
<td>Disabled</td>
</tr>
<tr>
<td>- SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input.</td>
<td>0</td>
<td>Enabled</td>
<td></td>
</tr>
<tr>
<td>- SPDIFRX in STATE_IDLE or,</td>
<td>1</td>
<td>0</td>
<td>Disabled</td>
</tr>
<tr>
<td>- SPDIFRX in STATE_STOP or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- SPDIFRX did not complete the fine synchronization (on-going)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- SPDIFRX is in STATE_RCV, but no data (transitions) detected on the selected SPDIFRX input.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 57.3.12 DMA interface

The SPDIFRX interface is able to perform communication using the DMA.

**Note:** The user must refer to product specifications for availability of the DMA controller.

The SPDIFRX offers two independent DMA channels:
- A DMA channel dedicated to the data transfer
- A DMA channel dedicated to the channel status and user data transfer

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register without CPU intervention.

For the use of DMA for the control data refer to Section 57.3.8: Dedicated control flow.

#### Table 603. Conditions of spdifrx_symb_ck generation (continued)

<table>
<thead>
<tr>
<th>SPDIFRX states and conditions</th>
<th>CKSEN</th>
<th>CKSBKPEN</th>
<th>spdifrx_symb_ck state</th>
</tr>
</thead>
<tbody>
<tr>
<td>– SPDIFRX in STATE_IDLE, but with valid values for WIDTH40 and WIDTH24 oraldi with invalid values for WIDTH40 and WIDTH24 or,</td>
<td></td>
<td></td>
<td>Enabled</td>
</tr>
<tr>
<td>– SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = ‘1’) or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– SPDIFRX in STATE_SYNC the on-going fine synchronization is not completed, but WIDTH40 and WIDTH24 contain the valid values from the previous synchronization or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– SPDIFRX in STATE_STOP, but with valid values for WIDTH40 and WIDTH24.</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>– SPDIFRX in IDLE, with invalid values for WIDTH40 and WIDTH24 or,</td>
<td></td>
<td></td>
<td>Disabled</td>
</tr>
<tr>
<td>– SPDIFRX in STOP with invalid values for WIDTH40 and WIDTH24 (SERR = ‘1’) or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– SPDIFRX in STATE_SYNC with invalid values for WIDTH40 and WIDTH24, and did not completed the on-going fine synchronization or,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>– SPDIFRX in STATE_RCV and no transitions detected on the selected SPDIFRX input</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that when the flag SERR is set to ‘1’, neither the symbol clock nor the backup clock can be generated, since there is no synchronization.

Note that when both CKSEN and CKSBKPEN are set to ‘1’, the symbol clock looses some transitions when the SPDIFRX switches from STATE_SYNC or STATE_RCV to STATE_STOP, or STATE_IDLE.

The bits CKSEN and CKSBKPEN are located into SPDIFRX control register (SPDIFRX_CR).
57.3.13 Interrupt generation

An interrupt line is shared between:

- Reception events for data flow (RXNE)
- Reception event for control flow (CSRNE)
- Data corruption detection (PERR)
- Transfer flow interruption (OVR)
- Frame structure and synchronization errors (SERR, TERR and FERR)
- Start of new block interrupt (SBD)
- Synchronization done (SYNCD)

![Figure 845. SPDIFRX interface interrupt mapping diagram](MSv35928V3)

**Clearing interrupt source**

- RXNE is cleared when SPDIFRX_FMTx_DR register is read
- CSRNE is cleared when SPDIFRX_CSR register is read
- FERR is cleared when SPDIFRXEN is set to 0
- SERR is cleared when SPDIFRXEN is set to 0
- TERR is cleared when SPDIFRXEN is set to 0
- Others are cleared through SPDIFRX_IFCR register

**Note:**

*The SBD event can only occur when the SPDIFRX is synchronized to the input stream (SYNCD = 1).

The SBD flag behavior is not guaranteed when the sub-frame which contains the B preamble is lost due to an overrun.*
57.3.14 Register protection

The SPDIFRX block embeds some hardware protection to avoid erroneous use of control registers. The table hereafter shows the bit field properties according to the SPDIFRX state.

Table 604. Bit field property versus SPDIFRX state

| Registers | Field   | SPDIFRXEN
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>00 (STATE_IDLE)</td>
<td>01 (STATE_SYNC)</td>
</tr>
<tr>
<td>SPDIFRX_CR</td>
<td>INSEL</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>WFA</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>NBTR</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>CHSEL</td>
<td>rw</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>CBDMAEN</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>PTMSK</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>CUMSK</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>VMSK</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>PMSK</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>DRFMT</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>RXSTEO</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td>RXDMAEN</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>SPDIFRX_IMR</td>
<td>All fields</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

The table clearly shows that fields such as INSEL must be programmed when the SPDIFRX is in STATE_IDLE. In the others SPDIFRX states, the hardware prevents writing to this field.

Note: Even if the hardware allows the writing of CBDMAEN and RXDMAEN “on-the-fly”, it is not recommended to enable the DMA when the SPDIFRX already receives data.

Each of the mask bits (such as PMSK, VMSK) can be changed “on-the-fly” at any SPDIFRX state, but any change does not affect data which are already hold in SPDIFRX_FMTx_DR.

57.4 Programming procedures

The following example illustrates a complete activation sequence of the SPDIFRX block. The data path and channel status and user information both use a dedicated DMA channel. The activation sequence is then split into the following steps:

- Wait for valid data on the selected SPDIFRX_IN input
- Synchronize to the S/PDIF stream
- Read the channel status and user information in order to setup the complete audio path
- Start data acquisition
A simple way to check if valid data are available into the SPDIFRX_IN line is to switch the SPDIFRX into the STATE_SYNC, with bit WFA set to 1. The description hereafter focuses on detection. It is also possible to implement this function as follows:

- The software has to check from time to time (i.e. every 100 ms for example) if the SPDIFRX can find synchronization. This can be done by checking if the bit TERR is set. When it is set it indicates that no activity as been found.
- Connect the SPDIFRX_IN input to an external interrupt event block in order to detect transitions of SPDIFRX_IN line. When activity is detected, then SPDIFRXEN can be set to 01 or 11.

For those two implementations, the bit WFA is set to 0.

### 57.4.1 Initialization phase

- The initialization function looks like this:
- Configure the DMA transfer for both audio samples and IEC60958 channel status and user information (DMA channel selection and activation, priority, number of data to transfer, circular/no circular mode, DMA interrupts)
- Configure the destination address:
  - Configure the address of the SPDIFRX_CSR register as source address for IEC60958 channel status and user information
  - Configure the address of the SPDIFRX_FMTx_DR register as source address for audio samples
  - Enable the generation of the spdifrx_ker_ck. Refer to Table 602 in order to define the minimum clock frequency versus supported audio sampling rate. Note that the audio sampling rate of the received stream is not known in advance. This means that the user has to select a spdifrx_ker_ck frequency at least 704 times higher than the maximum audio sampling rate the application is supposed to handle: for example if the application is able to handle streams to up to 96 kHz, then F_{spdifrx_ker_ck} must be at least 704 x 96 kHz = 67.6 MHz
- Enable interrupt for errors and event signaling (IFEIE = SYNCDIE = OVRIE, PERRIE = 1, others set to 0). Note that SYNCDIE can be set to 0.
- Configure the SPDIFRX_CR register:
  - INSEL must select the wanted input
  - NBTR = 2, WFA = 1 (16 re-tries allowed, wait for activity before going to synchronization phase),
  - PTMSK = CUMSK = 1 (Preamble, C and U bits are not mixed with data)
  - VMSK = PMSK = 0 (Parity error and validity bit mixed with data)
  - CHSEL = 0 (channels status are read from sub-frame A)
  - DRFMT = 01 (data aligned to the left)
  - RXSTEO = 1 (expected stereo mode linear)
  - CBDMAEN = RXDMAEN = 1 (enable DMA channels)
  - SPDIFRXEN = 01 (switch SPDIFRX to STATE_SYNC)
- The CPU can enter in WFI mode

Then the CPU receives interrupts coming either from DMA or SPDIFRX.
57.4.2 Handling of interrupts coming from SPDIFRX

When an interrupt from the SPDIFRX is received, then the software has to check what is the source of the interrupt by reading the SPDIFRX_SR register.

- If SYNCD is set to 1, then it means that the synchronization is properly completed. No action has to be performed in our case as the DMA is already programmed. The software just needs to wait for DMA interrupt in order to read channel status information.
  The SYNCD flag must be cleared by setting SYNCDCF bit of SPDIFRXIFn register to 1.

- If TERR or SERR or FERR are set to 1, the software has to set SPDIFRXEN to 0, and re-start from the initialization phase.
  - TERR indicates that a time-out occurs either during synchronization phase or after.
  - SERR indicates that the synchronization fails because the maximum allowed re-tries are reached.
  - FERR indicates that the reading of information after synchronization fails (such as unexpected preamble, bad data decoding).

- If PERR is set to 1, it means that a parity error is detected, so one of the received audio sample or the channel status or user data bits are corrupted. The action taken here depends on the application: one action can be to drop the current channel status block as it is not reliable. There is no need to re-start from the initialization phase, as the synchronization is not lost.
  The PERR flag must be cleared by setting PERRCF bit of SPDIFRXIFn register to 1.

57.4.3 Handling of interrupts coming from DMA

If an interrupt comes from the DMA channel used of the channel status (SPDIFRX_CSR):

If no error occurred (that is PERR), the CPU can start the decoding of channel information. For example bit 1 of the channel status informs the user if the current stream is linear or not. This information is very important in order to set-up the proper processing chain. In the same way, bits 24 to 27 of the channel status give the sampling frequency of the stream incoming stream.

Thanks to that information, the user can then configure the RXSTEO bit and DRFMT field prior to start the data reception. For example if the current stream is non linear PCM then RXSTEO is set to 0, and DRFMT is set to 10. Then the user can enable the data reception by setting SPDIFRXEN to 11.

The SOB bit, when set to 1 indicates the start of a new block. This information helps the software to identify the bit 0 of the channel status. Note that if the DMA generates an interrupt every time 24 values are transferred into the memory, then the first word always corresponds to the start of a new block.

If an interrupt comes from the DMA channel used of the audio samples (SPDIFRX_FMTx_DR):

The process performed here depends of the data type (linear or non-linear), and on the data format selected.

For example in linear mode, if PE or V bit is set a special processing can be performed locally in order to avoid spurs on output. In non-linear mode those bits are not important as data frame have their own checksum.
57.5 **SPDIFRX interface registers**

57.5.1 **SPDIFRX control register (SPDIFRX_CR)**

Only 32-bit accesses are allowed in this register.

Address offset: 0x00

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>bits</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td><strong>CKSBKPEN</strong>: Backup symbol clock enable</td>
</tr>
<tr>
<td></td>
<td>This bit is set/reset by software.</td>
</tr>
<tr>
<td></td>
<td>1: The SPDIFRX generates a backup symbol clock if CKSEN = 1.</td>
</tr>
<tr>
<td></td>
<td>0: The SPDIFRX does not generate a backup symbol clock.</td>
</tr>
<tr>
<td>20</td>
<td><strong>CKSEN</strong>: Symbol clock enable</td>
</tr>
<tr>
<td></td>
<td>This bit is set/reset by software.</td>
</tr>
<tr>
<td></td>
<td>1: The SPDIFRX generates a symbol clock.</td>
</tr>
<tr>
<td></td>
<td>0: The SPDIFRX does not generate a symbol clock.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18-16</td>
<td><strong>INSEL[2:0]</strong>: SPDIFRX input selection(1)</td>
</tr>
<tr>
<td></td>
<td>000: SPDIFRX_IN0 selected</td>
</tr>
<tr>
<td></td>
<td>001: SPDIFRX_IN1 selected</td>
</tr>
<tr>
<td></td>
<td>010: SPDIFRX_IN2 selected</td>
</tr>
<tr>
<td></td>
<td>011: SPDIFRX_IN3 selected</td>
</tr>
<tr>
<td></td>
<td>other: reserved</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td><strong>WFA</strong>: Wait for activity(1)</td>
</tr>
<tr>
<td></td>
<td>This bit is set/reset by software.</td>
</tr>
<tr>
<td></td>
<td>1: The SPDIFRX waits for activity on SPDIFRX_IN line (4 transitions) before performing the synchronization.</td>
</tr>
<tr>
<td></td>
<td>0: The SPDIFRX does not wait for activity on SPDIFRX_IN line before performing the synchronization.</td>
</tr>
</tbody>
</table>
Bits 13:12  **NBTR[1:0]**: Maximum allowed re-tries during synchronization phase\(^{(1)}\)
- 00: No re-try is allowed (only one attempt)
- 01: 3 re-tries allowed
- 10: 15 re-tries allowed
- 11: 63 re-tries allowed

Bit 11  **CHSEL**: Channel selection\(^{(1)}\)
This bit is set/reset by software.
- 1: The control flow takes the channel status from channel B.
- 0: The control flow takes the channel status from channel A.

Bit 10  **CBDMAEN**: Control buffer DMA enable for control flow\(^{(1)}\)
This bit is set/reset by software.
- 1: DMA mode is enabled for reception of channel status and used data information.
- 0: DMA mode is disabled for reception of channel status and used data information.

**Note**: When this bit is set, the DMA request is made whenever the CSRNE flag is set.

Bit 9  **PTMSK**: Mask of preamble type bits\(^{(1)}\)
This bit is set/reset by software.
- 1: The preamble type bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
- 0: The preamble type bits are copied into the SPDIFRX_FMTx_DR.

Bit 8  **CUMSK**: Mask of channel status and user bits\(^{(1)}\)
This bit is set/reset by software.
- 1: The channel status and user bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
- 0: The channel status and user bits are copied into the SPDIFRX_FMTx_DR.

Bit 7  **VMSK**: Mask of validity bit\(^{(1)}\)
This bit is set/reset by software.
- 1: The validity bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
- 0: The validity bit is copied into the SPDIFRX_FMTx_DR.

Bit 6  **PMSK**: Mask parity error bit\(^{(1)}\)
This bit is set/reset by software.
- 1: The parity error bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
- 0: The parity error bit is copied into the SPDIFRX_FMTx_DR.

Bits 5:4  **DRFMT[1:0]**: RX data format\(^{(1)}\)
This bit is set/reset by software.
- 11: reserved
- 10: Data sample are packed by setting two 16-bit sample into a 32-bit word.
- 01: Data samples are aligned in the left (MSB)
- 00: Data samples are aligned in the right (LSB).
Bit 3  **RXSTEO**: Stereo mode\(^{(1)}\)

This bit is set/reset by software.
1: The peripheral is in stereo mode.
0: The peripheral is in mono mode.

*Note*: This bit is used in case of overrun situation in order to handle misalignment.

Bit 2  **RXDMAEN**: Receiver DMA enable for data flow\(^{(1)}\)

This bit is set/reset by software.
1: DMA mode is enabled for reception.
0: DMA mode is disabled for reception.

*Note*: When this bit is set, the DMA request is made whenever the RXNE flag is set.

Bits 1:0  **SPDIFRXEN[1:0]**: Peripheral block enable\(^{(1)}\)

This field is modified by software.

It must be used to change the peripheral phase among the three possible states: STATE_IDLE, STATE_SYNC and STATE_RCV.
00: Disable SPDIFRX (STATE_IDLE).
01: Enable SPDIFRX synchronization only.
10: Reserved
11: Enable SPDIF receiver.

*Note*: It is not possible to transition from STATE_RCV to STATE_SYNC, the user must first go the STATE_IDLE.

it is possible to transition from STATE_IDLE to STATE_RCV: in that case the peripheral transitions from STATE_IDLE to STATE_SYNC and as soon as the synchronization is performed goes to STATE_RCV.

1. Refer to Section 57.3.14: Register protection for additional information on fields properties.

### 57.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR)

Address offset: 0x04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IFE</th>
<th>SYNCD</th>
<th>SBLK</th>
<th>OVR</th>
<th>PERR</th>
<th>CSRNE</th>
<th>RXNE</th>
</tr>
</thead>
<tbody>
<tr>
<td>IE</td>
<td>IE</td>
<td>IE</td>
<td>IE</td>
<td>IE</td>
<td>IE</td>
<td>IE</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:7  Reserved, must be kept at reset value.

Bit 6  **IFEIE**: Serial interface error interrupt enable

This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever SERR = 1, TERR = 1, or FERR = 1 in the SPDIFRX_SR register.

Bit 5  **SYNCDIE**: Synchronization done

This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever SYNCD = 1 in the SPDIFRX_SR register.
57.5.3 **SPDIFRX status register (SPDIFRX_SR)**

Address offset: 0x08
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>WIDTH5[14:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **WIDTH5[14:0]**: duration of 5 symbols counted with spdifrx_ker_ck

This value represents the amount of spdifrx_ker_ck clock periods contained on a length of 5 consecutive symbols. This value can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of spdifrx_ker_ck.

For example if the spdifrx_ker_ck is fixed to 84 MHz, and WIDTH5 = 147d. The estimated sampling rate of the S/PDIF stream is:

\[
Fs = 5 \times \frac{F_{spdifrx_ker_ck}}{WIDTH5 \times 64} \approx 44.6 \text{ kHz},
\]
so the closest standard sampling rate is 44.1 kHz.

Note that WIDTH5 is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:9 Reserved, must be kept at reset value.
Bit 8  **TERR**: Time-out error
This bit is set by hardware when the counter TRCNT reaches its max value. It indicates that the time interval between two transitions is too long. It generally indicates that there is no valid signal on SPDIFRX_IN input.
This flag is cleared by writing SPDIFRXEN to 0.
An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.
0: No sequence error is detected.
1: Sequence error is detected.

Bit 7  **SERR**: Synchronization error
This bit is set by hardware when the synchronization fails due to amount of re-tries for NBTR.
This flag is cleared by writing SPDIFRXEN to 0.
An interrupt is generated if IFEIE = 1 in the SPDIFRX_IMR register.
0: No synchronization error is detected.
1: Synchronization error is detected.

Bit 6  **FERR**: Framing error
This bit is set by hardware when an error occurs during data reception: such as preamble not at the expected place, short transition not grouped by pairs.
This is set by the hardware only if the synchronization is completed (SYNCD = 1).
This flag is cleared by writing SPDIFRXEN to 0.
An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.
0: No Manchester violation detected
1: Manchester violation detected

Bit 5  **SYNCD**: Synchronization done
This bit is set by hardware when the initial synchronization phase is properly completed.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
An interrupt is generated if SYNCDIE = 1 in the SPDIFRX_IMR register.
0: Synchronization is pending.
1: Synchronization is completed.

Bit 4  **SBD**: Synchronization block detected
This bit is set by hardware when a “B” preamble is detected.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
An interrupt is generated if SBLKIE = 1 in the SPDIFRX_IMR register.
0: No “B” preamble is detected.
1: “B” preamble is detected.

Bit 3  **OVR**: Overrun error
This bit is set by hardware when a received data is ready to be transferred in the SPDIFRX_FMTx_DR register while RXNE = 1 and both SPDIFRX_FMTx_DR and RX_BUF are full.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
An interrupt is generated if OVRIE=1 in the SPDIFRX_IMR register.
0: No overrun error
1: Overrun error is detected.

**Note**: When this bit is set, the SPDIFRX_FMTx_DR register content is not lost but the last data received are.
### Bit 2 PERR: Parity error
This bit is set by hardware when the data and status bits of the sub-frame received contain an odd number of 0 and 1.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
An interrupt is generated if PIE = 1 in the SPDIFRX_IMR register.
0: No parity error
1: Parity error

### Bit 1 CSRNE: Control buffer register not empty
This bit is set by hardware when a valid control information is ready.
This flag is cleared when reading SPDIFRX_CSR register.
An interrupt is generated if CBRDYIE = 1 in the SPDIFRX_IMR register.
0: No control word available on SPDIFRX_CSR register
1: A control word is available on SPDIFRX_CSR register.

### Bit 0 RXNE: Read data register not empty
This bit is set by hardware when a valid data is available into SPDIFRX_FMTx_DR register.
This flag is cleared by reading the SPDIFRX_FMTx_DR register.
An interrupt is generated if RXNEIE=1 in the SPDIFRX_IMR register.
0: Data is not received.
1: Received data is ready to be read.

### 57.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR)
Address offset: 0x0C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

### Bit 5 SYNCDCF: clears the synchronization done flag
Writing 1 in this bit clears the flag SYND in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

### Bit 4 SBDCEF: clears the synchronization block detected flag
Writing 1 in this bit clears the flag SBD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

### Bit 3 OVRCEF: clears the overrun error flag
Writing 1 in this bit clears the flag OVR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

### Bit 2 PERRCF: clears the parity error flag
Writing 1 in this bit clears the flag PERR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bits 1:0 Reserved, must be kept at reset value.
57.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR)

Address offset: 0x10
Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 00:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT[1:0]</td>
<td>C</td>
<td>U</td>
<td>V</td>
<td>PE</td>
<td>DR[23:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **PT[1:0]**: preamble type
These bits indicate the preamble received.
00: not used
01: Preamble B received
10: Preamble M received
11: Preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 27 **C**: channel status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 26 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 25 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 24 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

Bits 23:0 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR[23:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

57.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR)

Address offset: 0x10
Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 01:

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DR[23:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT[1:0]</td>
<td>C</td>
<td>U</td>
<td>V</td>
<td>PE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
SPDIF receiver interface (SPDIFRX)

57.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR)

Address offset: 0x10
Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT.
The data format proposed when DRFMT = 10, is dedicated to non-linear mode, as only 16 bits are used (bits 23 to 8 from S/PDIF sub-frame).

| Bit 3 | C: channel Status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0
| Bit 2 | U: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0
| Bit 1 | V: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0
| Bit 0 | PE: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

| Bits 31:16 | DRNL2[15:0]: data value
This field contains the channel A
| Bits 15:0 | DRNL1[15:0]: data value
This field contains the channel B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
### 57.5.8 SPDIFRX channel status register (SPDIFRX_CSR)

Address offset: 0x14
Reset value: 0x0000 0000

| Bit 31:25 | Reserved, must be kept at reset value. |
| Bit 24 | **SOB**: start of block |
| This bit indicates if the bit CS[0] corresponds to the first bit of a new block |
| 0: CS[0] is not the first bit of a new block |
| 1: CS[0] is the first bit of a new block |

| Bit 23:16 | **CS[7:0]**: channel A status information |
| Bit CS[0] is the oldest value |

| Bit 15:0 | **USR[15:0]**: user data information |
| Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B. |
| So USR[n] bits come from channel A is n is even, otherwise they come from channel B. |

### 57.5.9 SPDIFRX debug information register (SPDIFRX_DIR)

Address offset: 0x18
Reset value: 0x0000 0000

| Bit 31:29 | Reserved, must be kept at reset value. |
| Bit 28:16 | **TLO[12:0]**: threshold LOW (TLO = 1.5 x UI / T_{spdifrx_ker_ck}) |
| This field contains the current threshold LOW estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of TLO is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow: |
| Sampling Rate = [2 x TLO x T_{spdifrx_ker_ck} +/- T_{spdifrx_ker_ck}] x 2/3 |
| Note that TLO is updated by the hardware when SYNCD goes high, and then every frame. |

| Bit 15:13 | Reserved, must be kept at reset value. |
57.5.10 SPDIFRX interface register map

Table 605. SPDIFRX interface register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0x00   | SPDIFRX_CR    | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | Bits 12:0 THI[12:0]: threshold HIGH (THI = 2.5 x UI / T_{spdifrx_ker_ck})

This field contains the current threshold HIGH estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THI is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:

Sampling Rate = \([2 \times THI \times T_{spdifrx_ker_ck} \div T_{spdifrx_ker_ck}] \times 2/5\)

Note that THI is updated by the hardware when SYNCD goes high, and then every frame.

**Refer to Section 2.3 for the register boundary addresses.**
58 Management data input/output (MDIOS)

58.1 MDIOS introduction

An MDIO bus can be useful in systems where a master chip needs to manage (configure and get status data from) one or multiple slave chips. The bus protocol uses only two signals:

- MDC: the management data clock
- MDIO: the data line carrying the opcode (write or read), the slave (port) address, the MDIOS register address, and the data

In each transaction, the master either reads the contents of an MDIOS register in one of its slaves, or it writes data to an MDIOS register in one of its slaves.

The MDIOS peripheral serves as a slave interface to a MDIO bus. A MDIO master can use the MDC/MDIO lines to write and read 32 16-bit MDIOS registers, which are held in the MDIOS. These MDIOS registers are managed by the firmware. This allows the MDIO master to configure the application running on the STM32 and get status information from it.

The MDIOS can operate in Stop mode, optionally waking up the STM32 if the MDIO master performs a read or a write to one of its MDIOS registers.

58.2 MDIOS main features

The MDIOS includes the following features:

- 32 MDIOS register addresses, each of which is managed using separate input and output data registers:
  - 32 x 16-bit firmware read/write, MDIOS read-only output data registers
  - 32 x 16-bit firmware read-only, MDIOS write-only input data registers
- Configurable slave (port) address
- Independently maskable interrupts/events:
  - MDIOS register write
  - MDIOS register read
  - MDIOS protocol error
- Able to operate in and wake up from Stop mode
58.3 MDIOS functional description

58.3.1 MDIOS block diagram

![Figure 846. MDIOS block diagram](image)

58.3.2 MDIOS pins and internal signals

Table 606 lists the MDIOS inputs and output signals connected to package pins or balls, while Table 607 shows the internal PWR signals.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDIOS_MDC</td>
<td>Digital input</td>
<td>MDIO master clock</td>
</tr>
<tr>
<td>MDIOS_MDIO</td>
<td>Digital input/output</td>
<td>MDIO signal (opcode, address, input/output data)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mdios_wkup</td>
<td>Digital output</td>
<td>MDIOS wake-up signal</td>
</tr>
<tr>
<td>mdios_it</td>
<td>Digital output</td>
<td>MDIOS interrupt signal</td>
</tr>
<tr>
<td>mdios_pclk</td>
<td>Digital input</td>
<td>APB clock</td>
</tr>
</tbody>
</table>

58.3.3 MDIOS protocol

The MDIOS protocol uses two signals:
- MDIOS_MDC: the clock, always driven by the master
- MDIOS_MDIO: signal carrying the opcode, address, and bidirectional data
Each transaction is performed using a frame, which contains 32 bits (each passed serially):

- 14 control bits, driven by the master
  - 2 start bits: always 01
  - 2 opcode bits: read = 10, write = 01
  - 5 port address bits, indicating which slave device is being addressed
  - 5 MDIOS register address bits, up to 32 MDIOS registers can be addressed in each slave
- 2 turnaround state bits
  - On write operations, the master drives 10
  - On read operations, the first bit is high-impedance, and the second bit is driven by the slave to 0
- 16 data bits
  - On write operations, data written to slave MDIOS register is driven by the master
  - On read operations, data read from slave MDIOS register is driven by the slave

Each frame is usually preceded by a preamble, where the MDIO stays at one for 32 MDC clocks. The master can continue to keep MDIO at one, indicating the idle condition, when it has no frame to send.

When MDIO signal is driven by the master, MDIOS samples it using the rising edge of MDC. When MDIOS drives MDIO, the output changes on the rising edge of MDC.

**Figure 847. MDIO protocol write frame waveform**

**Figure 848. MDIO protocol read frame waveform**

### 58.3.4 MDIOS enabling and disabling

The MDIOS is enabled by setting the EN bit in MDIOS_CR. When EN = 1, the MDIOS monitors the MDIO bus and service frames addressed to one of its MDIOS registers.

When the MDIOS is enabled (setting EN to one), the same write operation to MDIOS_CR must properly set the PORT_ADDRESS[4:0] field to indicate the slave port address. A frame is ignored by the MDIOS if its port address is not the same as PORT_ADDRESS[4:0] (presumably intended for another slave).
When EN = 0, the MDIOS ignores the frames being transmitted on the MDC/MDIO lines, and the IP is in a reduced consumption mode. Clearing EN also clears all of the DIN registers. If EN is cleared while the MDIOS is driving read data, it immediately releases the bus and does not drive the rest of the data. If EN is cleared while the MDIOS is receiving a frame, the frame is aborted and the data is lost.

When the MDIOS is enabled, disabled and reenabled, the status flags are not cleared. For a correct operation, the firmware must clear the status flag before reenabling the MDIOS.

58.3.5 MDIOS data

From the point of view of the MDIO master, there are 32 16-bit MDIOS registers in the MDIOS, which can be written and read. In reality, for each MDIOS register ‘x’ there are two sets of registers: MDIOS_DINRx and MDIOS_DOUTRx.

Input data

When the MDIO master transmits a frame, which writes to MDIOS register ‘x’ in the MDIOS, it is MDIOS_DINRx, which is updated with the incoming data. The registers MDIOS_DINR0 to MDIOS_DINR31 can be read by the firmware, but they can be written only by the MDIO master via the MDIO bus.

The contents of MDIOS_DINRx change immediately after the MDC rising edge when the last data bit is sampled.

If the firmware happens to read the contents of MDIOS_DINRx at the moment that it is being updated, there is a possibility that the value read is corrupted (a bit-by-bit cross between the old value and the new value). For this reason, the firmware must assure that two subsequent reads from the same MDIOS_DINRx give the same value and assure that the data is stable when it is read. In the very worst case, the firmware needs to read four times MDIOS_DINRx to:

1. Get the old value
2. Get an incoherent value (when reading at the moment the register changes)
3. Get the new value
4. Confirm the new value

If the firmware uses the WRF interrupt and can guarantee that it reads the MDIOS_DINRx register before any new MDIOS write frame completes, the firmware can perform a single read.

If the MDIO master performs a write operation with a register address, which is greater than 31, the MDIOS ignores the frame (the data is not saved and no flag is set).

Output data

When the MDIOS receives a frame, which requests to read register ‘x’, it returns the value found in MDIOS_DOUTRx. Thus, if the MDIO master expects to read the same value, which it previously wrote to MDIOS register ‘x’, the firmware must copy the data from MDIOS_DINRx to MDIOS_DOUTRx each time new data is written to MDIOS_DINRx. For correct operation, the firmware must copy the data to the MDIOS_DOUTRx register within a preamble (if the master sends preambles before each frame) plus 15 cycles time.

When an MDIOS register is read via the MDIO bus, the MDIOS passes the 16-bit value (from the corresponding MDIOS_DOUTRx) to the MDIOS clock domain during the 15th cycle of the read frame. If the firmware attempts to write MDIOS_DOUTRx while the MDIO
master is currently reading MDIOS register ‘x’, the firmware write operation is ignored if it occurs during the 15th cycle of the frame (during a one-MDC-cycle window). Therefore, after writing MDIOS_DOUTRx, the firmware must read back the same MDIOS_DOUTRx and confirm that the value was actually written. If MDIOS_DOUTRx does not contain the value, which was written, the firmware can simply try writing and rereading again.

If the MDIOS frequency is very slow compared to the mdios_pclk frequency, it is better not to tie up the CPU by continuously writing and rereading MDIOS_DOUTRx.

Note: The read flag (RDFx) is set as soon as the MDIOS_DOUTRx DOUT[15:0] value is passed to the MDIOS clock domain. Thus, when a write to MDIOS_DOUTRx is ignored (when the value read back is not the value, which was just written), the firmware can use a read interrupt to know when it is able to write MDIOS_DOUTRx.

Respect the following procedure if the MDC clock is very slow:
1. Write MDIOS_DOUTRx.
2. Assure that all of the read flags are zero (MDIOS_RDFR = 0x0000). Clear the flags if necessary using MDIOS_CRDFR.
3. Read back the same MDIOS_DOUTRx and compare the value with the value, which was written in step 1.
4. If the values are the same, the procedure is done. Otherwise, continue to step 5.
5. Enable read interrupts by setting the RDIE bit in MDIOS_CR1.
6. In the interrupt routine, assure that RDFx is set (no other read flags are set before bit x).
7. If this maximum delay cannot be guaranteed, go back to step 1.

If the MDIO master performs a read operation with a register address, which is greater than 31, the MDIOS returns a data value of all zeros.

58.3.6 MDIOS APB frequency
Whenever the firmware reads from MDIOS_DINRx or writes to MDIOS_DOUTRx, the frequency of the APB bus must be at least 1.5 times the MDC frequency. For example, if MDC is at 20 MHz, the APB must be at 30 MHz or faster.

58.3.7 Write/read flags and interrupts
When MDIOS register ‘x’ is written via the MDIO bus, the WRFx bit in MDIOS_WRFR is set. WRFx becomes one at the moment that MDIOS_DINRx is updated, which is when the last data bit is sampled on a write frame. An interrupt is generated if WRIEN = 1 (in MDIOS_CR). WRFx is cleared by software by writing 1 to CWRFx (in MDIOS_CWRFR).
When MDIOS register ‘x’ is read via the MDIO bus, the RDFx bit in MDIOS_RDFR is set. RDFx becomes one at the moment that MDIOS_DOUTRx is copied to the MDC clock domain, which is on the 15th cycle of a read frame. An interrupt is generated if RDIEN = 1 (in MDIOS_CR). RDFx is cleared by software by writing 1 to CRDFx (in MDIOS_CRDFR).
58.3.8 MDIOS error management

There are three types of errors with their corresponding error flags:

- Preamble error: PERF (bit 0 of MDIOS_SR)
- Start error: SERF (bit 1 of MDIOS_SR)
- Turnaround error: TERF (bit 2 of MDIOS_SR)

Each error flag is set by hardware when the corresponding error condition occurs. Each flag can be cleared by writing ‘1’ to the corresponding bit in the clear flag register (MDIOS_CLRFR).

An interrupt occurs if any of the three error flags is set while EIE = 1 (MDIOS_CR).

Besides, setting an error flag, the MDIOS performs no action for a frame in which an error is detected: MDIOS_DINRx are not updated and the MDIO line is not forced during the data phase.

For a given frame, errors do not accumulate. For example, if a preamble error is detected, no check is done for a start error or a turnaround error for the rest of the current frame.

When DPC = 0, following a detected error, all new frames, and errors are ignored until a complete full preamble has been detected.

When DPC = 1 (disable preamble check, MDIOS_CR[7]), all frames and new errors are ignored as long as one of the error flags is set. As soon as the error bit is cleared, the MDIOS starts looking for a start sequence. Thus, the application must clear the error flag only when it is sure that no frame is currently in progress. Otherwise, the MDIOS likely misinterprets the bits being sent and becomes desynchronized with the master.

Preamble errors

A preamble error occurs when a start sequence begins (with MDIO sampled at 0) without being immediately preceded by a preamble (MDIO sampled at 1 for at least 32 consecutive clocks).

Preamble errors are not reported after the MDIOS is first enabled (EN = 1 in MDIOS_CR) until after a full preamble is received. This is to avoid an error condition when the peripheral frame detection is enabled while a preamble or frame is already in progress. In this case, the MDIOS ignores the first frame (since it did not first detect a full preamble), but does not set PERF.

If the DPC bit (disable preamble check, MDIOS_CR[7]) is set, the MDIO master can send frames without preceding preambles and no preamble error is signaled. When DPC = 1, the application must assure that the master is not in the process of sending a frame at the moment that the MDIOS is enabled (EN is set). Otherwise, the slave might become desynchronized with the master.

Start errors

A start error occurs when an illegal start sequence occurs or if an illegal command is given. The start sequence must always be 01, and the command must be either 01 (write) or 10 (read).

As with preamble errors, start errors are not reported until after a full preamble is received.
Turnaround errors

A turnaround error occurs when an error is detected in the turnaround bits of write frames. The 15th bit of the write frame must be 1 and the 16th bit must be 0.

Turnaround errors are only reported after a full preamble is received, there is no start error, the port address in the current frame matches, and the register address is in the supported range 0 to 31.

58.3.9 MDIOS in Stop mode

The MDIOS can operate in Stop mode, responding to all reads, performing all writes, and causing the STM32 to wake up from Stop mode on MDIOS interrupts.

58.3.10 MDIOS interrupts

There is a single interrupt vector for the three types of interrupts (write, read, and error). Any of these interrupt sources can wake the STM32 up from Stop mode. All interrupt flags need to be cleared in order to clear the interrupt line.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write interrupt</td>
<td>WRF[31:0]</td>
<td>WRIE</td>
</tr>
<tr>
<td>Read interrupt</td>
<td>RDF[31:0]</td>
<td>RDIE</td>
</tr>
<tr>
<td>Error interrupt</td>
<td>PERF (preamble), SERF (start), TERF (turnaround)</td>
<td>EIE</td>
</tr>
</tbody>
</table>

Table 608. Interrupt control bits

58.4 MDIOS registers

58.4.1 MDIOS configuration register (MDIOS_CR)

Address offset: 0x000
Reset value: 0x0000 0000

| Bits 31:13 | Reserved, must be kept at reset value. |
| Bits 12:8  | PORT_ADDRESS[4:0]: slave address     |
|           | Can be written only when the peripheral is disabled (EN = 0). |
|           | If the address given by the MDIO master matches PORT_ADDRESS[4:0], the MDIOS services the frame. Otherwise, the frame is ignored. |
Bit 7 **DPC**: disable preamble check
- 0: MDIO master must give preamble before each frame.
- 1: MDIO master can send each frame without a preceding preamble, and the MDIOS does not signal a preamble error.

*Note:* When this bit is set, the application must be sure that no frame is currently in progress when the MDIOS is enabled. Otherwise, the MDIOS can become desynchronized with the master.

This bit cannot be changed unless **EN** = 0 (though it can be changed at the same time that **EN** is being set).

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **EIE**: error interrupt enable
- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

*Note:* When this bit is set, an interrupt is generated if any of the error flags (PERF, SERF, or TERF in the MDIOS_SR register) is set.

Bit 2 **RDIE**: register read interrupt enable
- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

*Note:* When this bit is set, an interrupt is generated if any of the read flags (RDF[31:0] in the MDIOS_RDFR register) is set.

Bit 1 **WRIE**: register write interrupt enable
- 0: Interrupt is disabled.
- 1: Interrupt is enabled.

*Note:* When this bit is set, an interrupt is generated if any of the read flags (WRF[31:0] in the MDIOS_WRFR register) is set.

Bit 0 **EN**: peripheral enable
- 0: MDIOS is disabled.
- 1: MDIOS is enabled and monitoring the MDIO bus (MDC/MDIO).

### 58.4.2 MDIOS write flag register (MDIOS_WRFR)

**Address offset:** 0x004  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRF[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| WRF[15:0] |
| r | r | r | r | r | r | r | r | r | r | r | r | r | r | r |
### 58.4.3 MDIOS clear write flag register (MDIOS_CWRFR)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
<td>RC_W1</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0 WRF[31:0]:** Write flags for MDIOS registers 0 to 31.  
Each bit is set by hardware when the MDIO master performs a write to the corresponding MDIOS register. An interrupt is generated if WRIE (in MDIOS_CR) is set.  
Each bit is cleared by software by writing ‘1’ to the corresponding CWRF bit in the MDIOS_CWRFR register.  
For WRFx:  
0: MDIOS register x is not written by the MDIO master.  
1: MDIOS register x is written by the MDIO master and the data is available in DIN[15:0] in the MDIOS_DINRx register.

### 58.4.4 MDIOS read flag register (MDIOS_RDFR)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:0 RDF[31:0]:** Read flags for MDIOS registers 0 to 31.  
Each bit is set by hardware when the MDIO master performs a read from the corresponding MDIOS register. An interrupt is generated if RDIE (in MDIOS_CR) is set.  
Each bit is cleared by software by writing one to the corresponding CRDF bit in the MDIOS_CRDFR register.  
For RDFx:  
0: MDIOS register x is not read by the MDIO master.  
1: MDIOS register x is read by the MDIO master.
58.4.5 MDIOS clear read flag register (MDIOS_CRDFR)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

Bits 31:0 **CRDF[31:0]**: clear the read flag
Writing 1 to CRDFx clears the RDFx bit in the MDIOS_RDF register.

58.4.6 MDIOS status register (MDIOS_SR)

Address offset: 0x014
Reset value: 0x0000 0000

Writes to this register have no effect.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

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

Bit 2 **TERF**: turnaround error flag
0: No turnaround error has occurred.
1: A turnaround error has occurred.

*Note: Writing 1 to CTERF (MDIOS_CLRFR) clears this bit.*

Bit 1 **SERF**: start error flag
0: No start error has occurred.
1: A start error has occurred.

*Note: Writing 1 to CSERF (MDIOS_CLRFR) clears this bit.*

Bit 0 **PERF**: preamble error flag
0: No preamble error has occurred.
1: A preamble error has occurred.

*Note: Writing 1 to CPERF (MDIOS_CLRFR) clears this bit.*

*This bit is not set if DPC (disable preamble check, MDIOS_CR[7]) is set.*
58.4.7 MDIOS clear flag register (MDIOS_CLRFR)

Address offset: 0x018
Reset value: 0x0000 0000

Reads on this register returns all zeros.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

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

Bit 2 **CTERF**: clear the turnaround error flag
Writing 1 to this bit clears the TERF flag (in MDIOS_SR).
When DPC = 1 (MDIOS_CR[7]), the TERF flag must be cleared only when there is not a frame already in progress.

Bit 1 **CSERF**: clear the start error flag
Writing 1 to this bit clears the SERF flag (in MDIOS_SR).
When DPC = 1 (MDIOS_CR[7]), the SERF flag must be cleared only when there is not a frame already in progress.

Bit 0 **CPERF**: clear the preamble error flag
Writing 1 to this bit clears the PERF flag (in MDIOS_SR).

58.4.8 MDIOS input data register x (MDIOS_DINRx)

Address offset: 0x100 + 0x4 * x, (x = 0 to 31)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIN[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DIN[15:0]**: input data received from MDIO master during write frames
This field is written by hardware with the 16-bit data received in a write frame, which is addressed to MDIOS register x.
58.4.9  MDIOS output data register x (MDIOS_DOUTRx)

Address offset: 0x180 + 0x4 * x, (x = 0 to 31)
Reset value: 0x0000 0000

58.4.10  MDIOS register map

Table 609. MDIOS register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x004</td>
<td>MDIOS_WRFR</td>
<td>WRF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x008</td>
<td>MDIOS_CWRFR</td>
<td>CWRF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x00C</td>
<td>MDIOS_RDFR</td>
<td>RDF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x010</td>
<td>MDIOS_CRDFR</td>
<td>CRDF[31:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x014</td>
<td>MDIOS_SR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x018</td>
<td>MDIOS_CLRFR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C-</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>MDIOS_DINRx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x=0 to 31)</td>
<td>Last address:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x17C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.
Bits 15:0  DOUT[15:0]: output data sent to MDIO master during read frames
This field is written by software. These 16 bits are serially output on the MDIO bus during read frames, which address the MDIOS register x.
Refer to Section 2.3 for the register boundary addresses.

| Offset       | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x180 + 0x4 \* x (x=0 to 31) Last address: 0x1FC | MDIOS_DOUTRx |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|             | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
59 Secure digital input/output MultiMediaCard interface (SDMMC)

59.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (eMMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and eMMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed) (HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with SD memory card specifications version 6.0. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with SDIO card specification version 4.0. Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/eMMC card at any one time and a stack of eMMC.

59.2 SDMMC implementation

<table>
<thead>
<tr>
<th>SDMMC modes/features(1)</th>
<th>SDMMC1</th>
<th>SDMMC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable delay</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_CKIN</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC_CDIR, SDMMC_D0DIR</td>
<td>X</td>
<td>-</td>
</tr>
<tr>
<td>SDMMC_D123DIR</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>

1. X = supported.
59.3 **SDMMC bus topology**

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/eMMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:
- Block mode: data block(s) with block size $2^N$ bytes with $N$ in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- eMMC Stream mode: continuous data stream

Data transfers to/from eMMC cards are done in data blocks or streams.

**Figure 849. SDMMC “no response” and “no data” operations**

**Figure 850. SDMMC (multiple) block read operation**

*Note: The Stop Transmission command is not required at the end of a eMMC multiple block read with predefined block count.*
**Figure 851. SDMMC (multiple) block write operation**

Note: The Stop Transmission command is not required at the end of an e-MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

**Figure 852. SDMMC (sequential) stream read operation**

**Figure 853. SDMMC (sequential) stream write operation**

Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).
59.4 SDMMC operation modes

Table 611. SDMMC operation modes SD and SDIO

<table>
<thead>
<tr>
<th>SDIO bus speed modes$^{(1)(2)}$</th>
<th>Max bus speed $^{(3)}$ [Mbyte/s]</th>
<th>Max clock frequency $^{(4)}$ [MHz]</th>
<th>Signal voltage [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS (default speed)</td>
<td>12.5</td>
<td>25</td>
<td>3.3</td>
</tr>
<tr>
<td>HS (high speed)</td>
<td>25</td>
<td>50</td>
<td>3.3</td>
</tr>
<tr>
<td>SDR12</td>
<td>12.5</td>
<td>25</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR25</td>
<td>25</td>
<td>50</td>
<td>1.8</td>
</tr>
<tr>
<td>DDR50</td>
<td>50</td>
<td>50</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR50</td>
<td>50</td>
<td>100</td>
<td>1.8</td>
</tr>
<tr>
<td>SDR104</td>
<td>104</td>
<td>208</td>
<td>1.8</td>
</tr>
</tbody>
</table>

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 612. SDMMC operation modes e-MMC

<table>
<thead>
<tr>
<th>e-MMC bus speed modes $^{(1)(2)}$</th>
<th>Max bus speed $^{(3)}$ [Mbyte/s]</th>
<th>Max clock frequency $^{(4)}$ [MHz]</th>
<th>Signal voltage $^{(5)}$ [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy compatible</td>
<td>26</td>
<td>26</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed SDR</td>
<td>52</td>
<td>52</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed DDR</td>
<td>104</td>
<td>52</td>
<td>3/1.8/1.2V</td>
</tr>
<tr>
<td>High speed HS200</td>
<td>200</td>
<td>200</td>
<td>1.8/1.2V</td>
</tr>
</tbody>
</table>

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. e-MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.
59.5 SDMMC functional description

The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the eMMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

59.5.1 SDMMC block diagram

*Figure 854* shows the SDMMC block diagram.

![SDMMC block diagram](MSv39277V3)

59.5.2 SDMMC pins and internal signals

*Table 613* lists the SDMMC internal input/output signals, *Table 614* the SDMMC pins (alternate functions).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdmmc_ker_ck</td>
<td>Digital input</td>
<td>SDMMC kernel clock</td>
</tr>
<tr>
<td>sdmmc_hclk</td>
<td>Digital input</td>
<td>AHB clock</td>
</tr>
<tr>
<td>sdmmc_it</td>
<td>Digital output</td>
<td>SDMMC global interrupt</td>
</tr>
</tbody>
</table>
59.5.3 General description

The SDMMC_D[7:0] lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an eMMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The SDMMC_D0DIR signal indicates the I/O direction for the SDMMC_D0 data line, the SDMMC_D123DIR for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal SDMMC_CDIR.
SDMMC_CK clock to the card originates from `sdmmc_ker_ck`:

- When the `sdmmc_ker_ck` clock has 50% duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the `sdmmc_ker_ck` duty cycle is not 50%, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See Figure 855.

**Figure 855. SDMMC Command and data phase relation**

![Figure 855. SDMMC Command and data phase relation](image)

**Table 615. SDMMC Command and data phase selection**

<table>
<thead>
<tr>
<th>CLKDIV</th>
<th>DDR</th>
<th>NEGEDGE</th>
<th>SDMMC_CK</th>
<th>Command out</th>
<th>Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>x</td>
<td><code>sdmmc_ker_ck</code></td>
<td>Generated on <code>sdmmc_ker_ck</code> falling edge</td>
<td></td>
</tr>
<tr>
<td>&gt;0</td>
<td>0</td>
<td>0</td>
<td><code>sdmmc_ker_ck</code></td>
<td>Generated on <code>sdmmc_ker_ck</code> falling edge succeeding the SDMMC_CK rising edge.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td><code>sdmmc_ker_ck</code></td>
<td>Generated on the same <code>sdmmc_ker_ck</code> rising edge that generates the SDMMC_CK falling edge.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td><code>sdmmc_ker_ck</code></td>
<td>Generated on <code>sdmmc_ker_ck</code> falling edge succeeding the SDMMC_CK rising edge.</td>
<td>Generated on <code>sdmmc_ker_ck</code> falling edge succeeding a SDMMC_CK edge.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td><code>sdmmc_ker_ck</code></td>
<td>Generated on the same <code>sdmmc_ker_ck</code> rising edge that generates the SDMMC_CK falling edge.</td>
<td></td>
</tr>
</tbody>
</table>

By default, the `sdmmc_io_in_ck` feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between `sdmmc_io_in_ck` signal (DLYB input dlyb_in_ck) and `sdmmc_fb_ck` clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the `sdmmc_fb_ck` clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.
When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback
clock input can be selected to sample the receive data.

For an SD/SDIO/eMMC card, the clock frequency can vary between 0 and 208 MHz
(limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on
SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit
per clock cycle.

59.5.4 SDMMC adapter

The SDMMC adapter (see Figure 854: SDMMC block diagram) is a multimedia/secure
digital memory card bus master that provides an interface to a MultiMediaCard stack or to a
secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit,
command path and data transmit path use the SDMMC adapter clock domain
(sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter
feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the
sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter,
to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is
required for the SDMMC to support the SDR104 and HS200 operating mode and optional
for SDR50 and DDR50 modes.

Adapter register block

The adapter register block contains all system control registers, the SDMMC command and
response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC
Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in Figure 856, contains the power management functions, the
SDMMC_CK clock management with divider, and the I/O direction management.
The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:
- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.
Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.
- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the
CMDSENT flag is generated immediately after the command end bit. The RESPCMDR and RESPxR registers are not modified.
- If a command response is expected (WAITRESP ≠ 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  - When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
    a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
    b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
    The RESPCMDR and RESPxR registers are not modified.
  - When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
    a) When a start bit is detected the CPSM moves to the Receive state.
    b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by itself and on detecting the start bit the CPSM move to the Receive state.

- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  - When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
    When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  - When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
    The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  - When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
    The RESPCMDR and RESPxR registers are updated with received response.

- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  - When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  - When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the
DPSTM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

- When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  - BOO TMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  - BOO TMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

**Note:**
The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the \( N_{CC} \) and \( N_{RC} \) timing constraints. \( N_{CC} \) is the minimum delay between two host commands, and \( N_{RC} \) is the minimum delay between the host command and the card response.
The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for e\(^*\)MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and e\(^*\)MMC cards is shown in Table 616.

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

**Table 616. Command token format**

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>1</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>x</td>
<td>CRC7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various
coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and eMMC cards are shown in Table 617, Table 618 and Table 619.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

### Table 617. Short response with CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index (or reserved 111111)</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>x</td>
<td>CRC7</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

### Table 618. Short response without CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>46</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[45:40]</td>
<td>6</td>
<td>x</td>
<td>Command index (or reserved 111111)</td>
</tr>
<tr>
<td>[39:8]</td>
<td>32</td>
<td>x</td>
<td>Argument</td>
</tr>
<tr>
<td>[7:1]</td>
<td>7</td>
<td>1111111</td>
<td>(reserved 111111)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

### Table 619. Long response with CRC token format

<table>
<thead>
<tr>
<th>Bit position</th>
<th>Width</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>135</td>
<td>1</td>
<td>0</td>
<td>Start bit</td>
</tr>
<tr>
<td>134</td>
<td>1</td>
<td>0</td>
<td>Transmission bit</td>
</tr>
<tr>
<td>[133:128]</td>
<td>6</td>
<td>1111111</td>
<td>Reserved</td>
</tr>
<tr>
<td>[127:1]</td>
<td>127:8</td>
<td>x</td>
<td>CID or CSD slices</td>
</tr>
<tr>
<td></td>
<td>7:1</td>
<td>x</td>
<td>CRC7 (included in CID or CSD)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>End bit</td>
</tr>
</tbody>
</table>

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the
SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see Figure 855.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

\[
\text{CRC}[6:0] = \text{remainder } [(M(x) \times x^7) / G(x)]
\]

\[
G(x) = x^7 + x^3 + 1
\]

\[
M(x) = \text{(first bit)} \times x^n + \text{(second bit)} \times x^{n-1} + \ldots + \text{(last bit before CRC)} \times x^0
\]

Where \( n = 39 \) or \( 119 \).

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see Table 620.

<table>
<thead>
<tr>
<th>VSWWITCH</th>
<th>BOOTEN</th>
<th>BOOTMOD</th>
<th>CMDTRAN</th>
<th>WAITPEND</th>
<th>CMDSTOP</th>
<th>WAITINT</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Start voltage switch sequence</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>Start normal boot</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>Start alternative boot</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>Stop alternative boot.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>x</td>
<td></td>
<td></td>
<td>Send command with associated data transfer.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>eMMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>eMMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>x</td>
<td>Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>Enter eMMC wait interrupt (Wait-IRQ) mode.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Any other none specific command</td>
</tr>
</tbody>
</table>
The command/response path implements the status flags and associated clear bits shown in Table 621:

### Table 621. Command path status flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMDSENT</td>
<td>Set at the end of the command without response (CPSM moves from Send to Idle).</td>
</tr>
<tr>
<td>CMDREND</td>
<td>Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).</td>
</tr>
<tr>
<td>CCRCFAIL</td>
<td>Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).</td>
</tr>
<tr>
<td>CTIMEOUT</td>
<td>Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).</td>
</tr>
<tr>
<td>CKSTOP</td>
<td>Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).</td>
</tr>
<tr>
<td>VSWEND</td>
<td>Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).</td>
</tr>
<tr>
<td>CPSMACT</td>
<td>Command transfer in progress (CPSM not in Idle state).</td>
</tr>
</tbody>
</table>

The command path error handling is shown in Table 622:

### Table 622. Command path error handling

<table>
<thead>
<tr>
<th>Error</th>
<th>CPSM state</th>
<th>Cause</th>
<th>Card action</th>
<th>Host action</th>
<th>CPSM action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Wait</td>
<td>No start bit in time</td>
<td>Unknown</td>
<td>Reset or cycle power card(^{(1)})</td>
<td>Move to Idle</td>
</tr>
<tr>
<td>CRC status</td>
<td>Receive</td>
<td>Negative status</td>
<td>Command ignored</td>
<td>Resend command(^{(1)})</td>
<td>Move to Idle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Transmission error</td>
<td>Command accepted</td>
<td>Resend command(^{(1)})</td>
<td></td>
</tr>
</tbody>
</table>

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

### Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. Figure 859 shows the data path block diagram.
The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)
In DDR mode, data is sampled on both edges of the SDMMC_CK according to the following rules, see also Figure 860 and Figure 861:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

**Figure 860. DDR mode data packet clocking**

**Figure 861. DDR mode CRC status / boot acknowledgment clocking**

**Data path state machine (DPSM)**

Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.
For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in Figure 862. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

**Figure 862. Data path state machine (DPSM)**

- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:
  
  `- When the data direction bit (DTDIR) indicates send, moves to the Wait_S.
  `- When the data direction bit (DTDIR) indicates receive, moves to the - Wait_R when BOOTACKEN register bit is clear.
  `- Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.
Note: **DTEN must not be used to start data transfer with SD, SDIO and eMMC cards.**

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.

- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    - If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    - If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.
- When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.
  - When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  - The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  - If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.

- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.
    a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
       - Wait_R state otherwise.
    b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
       - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
       - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
    - When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
    - When an CPSM Abort signal is received:
      - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
      - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
    - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.
• **Wait_S state:** the data path waits for data to be available from the FIFO.
  – If the data counter DATACOUNT > 0, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the Send state.
  – If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
    - When DTHOLD is disabled, the DATAEND flag is set.
    - When DTHOLD is enabled, the DHOLD flag is set.
  – When DTHOLD is set and the DATACOUNT > 0
    - When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
    - When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.
  – When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.
  – When receiving the CPSM Abort signal
    - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
    - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

**Note:** The DPSM remains in the Wait_S state for at least two clock periods to meet the NWR timing requirements, where NWR is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

• **Send state:** the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  – In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
    Before sending the last stream byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).
    - If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
    - When receiving the CPSM Abort signal
      - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
      - If the CPSM Abort signal is received during or after the 2 last bits of the transfer
with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate "non-erroneous transmission" or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates "erroneous transmission".
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.
The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period, the DTIMEOUT flag is set.

When DATATIME = 0:
- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

### Table 623. Data token format

<table>
<thead>
<tr>
<th>Description</th>
<th>Start bit</th>
<th>Data (^1)</th>
<th>CRC16</th>
<th>End bit</th>
<th>DTMODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block data</td>
<td>0</td>
<td>(DBLOCKSIZE, DATALENGTH)</td>
<td>Yes</td>
<td>1</td>
<td>00</td>
</tr>
<tr>
<td>SDIO multibyte</td>
<td>0</td>
<td>(DATALENGTH)</td>
<td>Yes</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>eMMC stream</td>
<td>0</td>
<td>(DATALENGTH)</td>
<td>No</td>
<td>1</td>
<td>10</td>
</tr>
</tbody>
</table>

\(^1\) The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in **Table 624:***

### Table 624. Data path status flags and clear bits

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAEND</td>
<td>Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)</td>
</tr>
<tr>
<td>RX</td>
<td>Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)</td>
</tr>
<tr>
<td>Boot</td>
<td></td>
</tr>
</tbody>
</table>
### Table 624. Data path status flags and clear bits (continued)

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DCRCFAIL</strong></td>
<td>TX: Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN</td>
</tr>
<tr>
<td></td>
<td>RX: Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)</td>
</tr>
<tr>
<td><strong>ACKFAIL</strong></td>
<td>Boot: Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)</td>
</tr>
<tr>
<td><strong>DTIMEOUT</strong></td>
<td>TX: Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.</td>
</tr>
<tr>
<td></td>
<td>RX: Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)</td>
</tr>
<tr>
<td><strong>ACKTIMEOUT</strong></td>
<td>Boot: Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)</td>
</tr>
<tr>
<td><strong>DBCKEND</strong></td>
<td>TX: When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from Busy to Wait_S)</td>
</tr>
<tr>
<td></td>
<td>RX: When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT &gt; 0). (DPSM moves from Receive to R_W)</td>
</tr>
<tr>
<td><strong>DHO_END</strong></td>
<td>TX: When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)</td>
</tr>
<tr>
<td></td>
<td>RX: When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT &gt;0). (DPSM moves from Wait_R to Idle)</td>
</tr>
<tr>
<td><strong>DABORT</strong></td>
<td>CMD: When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)</td>
</tr>
<tr>
<td></td>
<td>RX: When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)</td>
</tr>
<tr>
<td><strong>BUSYD0END</strong></td>
<td>CMD: Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)</td>
</tr>
<tr>
<td><strong>DPSMACT</strong></td>
<td>Data transfer in progress. (DPSM not in Idle state)</td>
</tr>
</tbody>
</table>
The data path error handling is shown in Table 625:

### Table 625. Data path error handling

<table>
<thead>
<tr>
<th>Error</th>
<th>DPSM state</th>
<th>Cause</th>
<th>Card action</th>
<th>Host action</th>
<th>DPSM action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout</td>
<td>Wait_Ack</td>
<td>No Ack in time</td>
<td>unknown</td>
<td>Card cycle power</td>
<td>Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timeout</td>
<td>Wait_R</td>
<td>No start bit in time</td>
<td>unknown</td>
<td>Stop data reception</td>
<td>Stop data reception Send stop transmission command</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Busy</td>
<td></td>
<td>Busy too long (due to data transfer)</td>
<td>unknown</td>
<td>Stop data reception Send stop transmission command</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>Busy</td>
<td></td>
<td>Busy too long (due to R1b)</td>
<td>unknown</td>
<td>Stop data reception Send stop transmission command</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>CRC</td>
<td>Receive</td>
<td>transmission error</td>
<td>Send further data</td>
<td>Stop data reception Send stop transmission command</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>CRC status</td>
<td>Busy</td>
<td>Negative status transmission error</td>
<td>Ignore further data</td>
<td>Stop data transmission Send stop transmission command</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>Ack status</td>
<td>Wait_Ack</td>
<td>transmission error</td>
<td>Send boot data</td>
<td>Stop boot procedure</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>Overrun</td>
<td>Receive</td>
<td>FIFO full</td>
<td>Send further data</td>
<td>Stop data reception Send stop transmission command</td>
<td>On CPSM Abort move to Idle</td>
</tr>
<tr>
<td>Underrun</td>
<td>Send</td>
<td>FIFO empty</td>
<td>Receive further data</td>
<td>Stop data transmission Send stop transmission command</td>
<td>On CPSM Abort move to Idle</td>
</tr>
</tbody>
</table>

**Data FIFO**

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contains a 32-bit wide, 16-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.
The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see Table 626.

### Table 626. Data FIFO access

<table>
<thead>
<tr>
<th>Data FIFO access</th>
<th>IDMAEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>From firmware via AHB slave interface</td>
<td>0</td>
</tr>
<tr>
<td>From IDMA via AHB master interface</td>
<td>1</td>
</tr>
</tbody>
</table>

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out.

The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   - For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in transmit mode (DTDIR = 0).
   - Configures the FIFO in transmit mode.
3. Enable the data transfer
   - either by sending a command from the CPSM with the CMDTRANS bit set
   - or by setting DTEN bit
4. When (DPSMACT = 1) write data to the FIFO.
   - The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.
- The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.

5. Write data to the FIFO.
   - When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   - When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.

6. When last data has been written wait for end of data (DATAEND flag)
   - SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in Table 627.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXFIFOF</td>
<td>Set to high when all transmit FIFO words contain valid data.</td>
</tr>
<tr>
<td>TXFIFOE</td>
<td>Set to high when the transmit FIFO does not contain valid data.</td>
</tr>
<tr>
<td>TXFIFOHE</td>
<td>Set to high when half or more transmit FIFO words are empty.</td>
</tr>
<tr>
<td>TXUNDERR</td>
<td>Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.</td>
</tr>
</tbody>
</table>

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.
The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   - For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Set the SDMMC in receive mode (DTDIR = 1).
   - Configures the FIFO in receive mode.
3. Enable the DPSM transfer
   - either by sending a command from the CPSM with the CMDTRANS bit set
   - or by setting DTEN bit.
4. When (DPSMACT = 1) the FIFO is ready to receive data.
   - The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
5. Read data from the FIFO.
   - When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   - When the FIFO is handled by the IDMA, the IDMA transfers the FIFO data.
6. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in Table 628.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RXFIFOF</td>
<td>Set to high when all receive FIFO words contain valid data</td>
</tr>
<tr>
<td>RXFIFOE</td>
<td>Set to high when the receive FIFO does not contain valid data.</td>
</tr>
<tr>
<td>RXFIFOHF</td>
<td>Set to high when half or more receive FIFO words contain valid data.</td>
</tr>
<tr>
<td>RXOVERR</td>
<td>Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.</td>
</tr>
</tbody>
</table>

**CLKMUX unit**

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.
The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.
The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.
The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

59.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according to DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according to DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.
59.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 8 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transferred. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode.
  - When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are transfered using single transfer mode.

- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transferred.
  - When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode.
  - When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer.
When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGTH, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transferred:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transferred (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

When a IDMA transfer error occurs (see Section : IDMA transfer error management) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see Section : IDMA transfer error management).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

**Figure 864. Linked list structures**

There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.
Whenever a linked list buffer has been transferred and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

**Linked list acknowledgment**

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see Section: IDMA transfer error management). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don’t care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

**IDMA transfer error management**

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transferred and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND):
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR, RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).
59.5.7 **AHB and SDMMC_CK clock relation**

The AHB must at least have 3x more bandwidth than the SDMMC bus bandwidth: for example, for SDR50 4-bit mode (50 Mbyte/s), the minimum sdmmc_hclk frequency is 37.5 MHz (150 Mbyte/s).

<table>
<thead>
<tr>
<th>SDMMC bus mode</th>
<th>SDMMC bus width</th>
<th>Maximum SDMMC_CK [MHz]</th>
<th>Minimum AHB clock [MHz]</th>
</tr>
</thead>
<tbody>
<tr>
<td>eMMC DS</td>
<td>8</td>
<td>26</td>
<td>19.5</td>
</tr>
<tr>
<td>eMMC HS</td>
<td>8</td>
<td>52</td>
<td>39</td>
</tr>
<tr>
<td>eMMC DDR52</td>
<td>8</td>
<td>52</td>
<td>78</td>
</tr>
<tr>
<td>eMMC HS200</td>
<td>8</td>
<td>200</td>
<td>150</td>
</tr>
<tr>
<td>SD DS / SDR12</td>
<td>4</td>
<td>25</td>
<td>9.4</td>
</tr>
<tr>
<td>SD HS / SDR25</td>
<td>4</td>
<td>50</td>
<td>18.8</td>
</tr>
<tr>
<td>SD DDR50</td>
<td>4</td>
<td>50</td>
<td>37.5</td>
</tr>
<tr>
<td>SD SDR50</td>
<td>4</td>
<td>100</td>
<td>37.5</td>
</tr>
<tr>
<td>SD SDR104</td>
<td>4</td>
<td>208</td>
<td>78</td>
</tr>
</tbody>
</table>

**Table 629. AHB and SDMMC_CK clock frequency relation**

59.6 **Card functional description**

59.6.1 **SD I/O mode**

The following features are SDMMC specific operations:
- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

<table>
<thead>
<tr>
<th>Operation mode</th>
<th>SDOEN</th>
<th>RWMOD</th>
<th>RWSTOP</th>
<th>RWSTART</th>
<th>DDIR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt detection</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Suspend/Resume operation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Read Wait SDMMC_CK clock stop (START)</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Read Wait SDMMC_CK clock stop (STOP)</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Read Wait SDMMC_D2 signaling (START)</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Read Wait SDMMC_D2 signaling (STOP)</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 630. SDIO special operation control**
SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in Table 630.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

![Figure 865. Asynchronous interrupt generation](image_url)

The timing of the interrupt period is depending on the bus speed mode.
In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

**Note:** DTEN must not be used to start data transfer with SD and eMMC cards.

---

**Figure 866. Synchronous interrupt period data read**

**Figure 867. Synchronous interrupt period data write**
In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.

- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.

- There is NO interrupt period at the data block gap.

*Note:* *DTEN must not be used to start data transfer with SD and eMMC cards.*

**Figure 868. Asynchronous interrupt period data read**
When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according Table 631.

Table 631. 4-bit mode Start, interrupt, and CRC-status Signaling detection

<table>
<thead>
<tr>
<th>SDMMC data line</th>
<th>Start</th>
<th>Interrupt</th>
<th>CRC-status</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC_D0</td>
<td>0</td>
<td>1 or CRC-status</td>
<td>0</td>
</tr>
<tr>
<td>SDMMC_D1</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_D2</td>
<td>0</td>
<td>1 or Read Wait</td>
<td>X</td>
</tr>
<tr>
<td>SDMMC_D3</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the eMMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.
To perform the suspend/resume operation on the bus, the host performs the following steps:
1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:
- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transferred at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after another operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transferred when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:
1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

*Note:* When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.
To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMD_SUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMD_SUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMD_SUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transferred.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

**SD I/O Read Wait**

There are two methods to pause the data transfer during the block gap:

1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

The SDMMC can perform a Read Wait with register settings according to Table 630.

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see Figure 870).
The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see Figure 871. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see Figure 872.
For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see Figure 873).

In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

### 59.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.
Table 632. CMD12 use cases

<table>
<thead>
<tr>
<th>Data operation</th>
<th>Stop Transmission command CMD12 Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC stream write</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC open ended multiple block write</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC block write with predefined block count</td>
<td>The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC stream read</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC open ended multiple block read</td>
<td>The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
<tr>
<td>SDMMC block read with predefined block count</td>
<td>The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.</td>
</tr>
</tbody>
</table>
All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   - This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   - The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit. The card when no data is being sent, does not start sending any new data.
   - The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit. The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   - When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST; this flushes the FIFO.
   - When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   - When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   - When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

### Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   - When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   - When DATALENGTH < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.
To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   - The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   - CMD12 stops the card sending data.

*Note:* The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

**Block operation and CMD12**

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   - The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   - CMD12 sets the card to Idle mode.
When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   - The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   - CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

*Note:* *The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.*

### 59.6.3 Sleep (CMD5)

The eMMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.
59.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):
- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):
- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:
1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.

2. Load CMD40 (GO_IRQ_STATE) in the command registers.

3. Enable wait for interrupt by setting WAITINT register bit.

4. Configure the CPSM to send a command immediately.
   - This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for an interrupt service request response.

5. To exit the wait for interrupt state (CPSM Wait state):
   - Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   - To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

59.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. If a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. Enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. When all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   - If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.
Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

To perform the alternative boot procedure the following steps needed:

1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. If a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. Enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. The reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - When the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. When all boot data has been received the DATAEND flag occurs.
   - When data CRC fails the DCRCFAIL flag is also generated.
   - When the data timeout occurs the DTIMEOUT flag is also generated.
9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.

10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.

11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

59.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 878. Command response R1b busy signaling

The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  - If BUSYD0 signals not busy, signal busy release to firmware
  - If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.
59.6.7 Reset and card cycle power

Reset

Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transferred. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transferred.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according Figure 879.

Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.
59.7 Hardware flow control

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see Section : IDMA transfer error management).

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a tOP and DtOP delay > 1 cycle, hardware flow control can not be used.

59.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch.
sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 881. CMD11 signal voltage switch sequence

To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.

2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.

3. The card returns an R1 response.  
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set. 
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.

4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.

5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   - When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.

6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.

7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.

8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].

9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
   – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
   – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

   The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by \(2^{12}\) cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

   The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by \(2^8\) cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

   The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.
To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

### 59.9 SDMMC interrupts
<table>
<thead>
<tr>
<th>Interrupt acronym</th>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
<th>Interrupt clear method</th>
<th>Exit from Sleep mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC</td>
<td>Command response CRC fail</td>
<td>CCRCFAIL</td>
<td>CCRCFAILIE</td>
<td>CCRCFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data block CRC fail</td>
<td>DCRCFAIL</td>
<td>DCRCFAILIE</td>
<td>DCRCFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command response timeout</td>
<td>CTIMEOUT</td>
<td>CTIMEOUTIE</td>
<td>CTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data timeout</td>
<td>DTIMEOUT</td>
<td>DTIMEOUTIE</td>
<td>DTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO underrun</td>
<td>TXUNDERR</td>
<td>TXUNDERRIE</td>
<td>TXUNDERRC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO overrun</td>
<td>RXOVERR</td>
<td>RXOVERRIE</td>
<td>RXOVERRC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command reponse received</td>
<td>CMDREN</td>
<td>CMDRENIE</td>
<td>CMDRENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer ended</td>
<td>DATAEND</td>
<td>DATAENDIE</td>
<td>DATAENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer hold</td>
<td>DHOLD</td>
<td>DHOLDIE</td>
<td>DHOLDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data block sent or received</td>
<td>DBCKEND</td>
<td>DBCKENDIE</td>
<td>DBCKENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Data transfer aborted</td>
<td>DABORT</td>
<td>DABORTIE</td>
<td>DABORTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO half empty</td>
<td>TXFIFOH</td>
<td>TXFIFOHIE</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO half full</td>
<td>RXFIFOH</td>
<td>RXFIFOHFIE</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO full</td>
<td>TXFIFO</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO full</td>
<td>RXFIFO</td>
<td>RXFIFOIE</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Transmit FIFO empty</td>
<td>TXFIFOE</td>
<td>TXFIFOEIE</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Receive FIFO empty</td>
<td>RXFIFOE</td>
<td>N/A</td>
<td>N/A</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Command response end of busy</td>
<td>BUSYD0END</td>
<td>BUSYD0ENDIE</td>
<td>BUSYD0ENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SDIO interrupt</td>
<td>SDIOIT</td>
<td>SDIOITIE</td>
<td>SDIOITC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Boot acknowledgment fail</td>
<td>ACKFAIL</td>
<td>ACKFAILIE</td>
<td>ACKFAILC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Boot acknowledgment timeout</td>
<td>ACKTIMEOUT</td>
<td>ACKTIMEOUTIE</td>
<td>ACKTIMEOUTC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>Voltage switch timing</td>
<td>VSWEND</td>
<td>VSWENDIE</td>
<td>VSWENDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SDMM_CK stopped in voltage switch</td>
<td>CKSTOP</td>
<td>CKSTOPIE</td>
<td>CKSTOPC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>IDMA transfer error</td>
<td>IDMATE</td>
<td>IDMATEIE</td>
<td>IDMATEC</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>IDMA buffer transfer complete</td>
<td>IDMABTC</td>
<td>IDMABTCIE</td>
<td>IDMABTCC</td>
<td>Yes</td>
</tr>
</tbody>
</table>
59.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHBM slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHBM bus error.

59.10.1 SDMMC power control register (SDMMC\_POWER)

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 4 DIRPOL: Data and command direction signals polarity selection
This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).
0: Voltage transceiver I/Os driven as output when direction signal is low.
1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 VSWITCHEN: Voltage switch procedure enable
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
This bit is used to stop the SDMMC\_CK after the voltage switch command response:
0: SDMMC\_CK clock kept unchanged after successfully received command response.
1: SDMMC\_CK clock stopped after successfully received command response.

Bit 2 VSWITCH: Voltage switch sequence start
This bit is used to start the timing critical section of the voltage switch sequence:
0: Voltage switch sequence not started and not active.
1: Voltage switch sequence started or active.

Bits 1:0 PWRCTRL[1:0]: SDMMC state control bits
These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).
These bits are used to define the functional state of the SDMMC signals:
00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC\_D[7:0], and SDMMC\_CMD are HiZ and SDMMC\_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are driven high.
01: Reserved (When written 01, PWRCTRL value does not change)
10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are driven low.
11: Power-on: the card is clocked, The first 74 SDMMC\_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC\_D[7:0], SDMMC\_CMD and SDMMC\_CK are controlled according the SDMMC operation. Any further write is ignored, PWRCTRL value keeps 11.
### 59.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004  
Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

<table>
<thead>
<tr>
<th>Bit 31:22</th>
<th>Description</th>
<th>Access</th>
<th>Bit 21:20</th>
<th>Description</th>
<th>Access</th>
<th>Bit 19</th>
<th>Description</th>
<th>Access</th>
<th>Bit 18</th>
<th>Description</th>
<th>Access</th>
<th>Bit 17</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
<td>rw</td>
<td>SELCLKRX[1:0]</td>
<td>Receive clock selection</td>
<td>rw</td>
<td>BUSSPEED</td>
<td>Bus speed for selection of SDMMC operating modes</td>
<td>rw</td>
<td>DDR</td>
<td>Data rate signaling</td>
<td>rw</td>
<td>HWFC_EN</td>
<td>Hardware flow control enable</td>
</tr>
<tr>
<td>00</td>
<td>sdmmc_io_in_ck selected as receive clock</td>
<td></td>
<td>01</td>
<td>SDMMC_CKIN feedback clock selected as receive clock</td>
<td></td>
<td>10</td>
<td>sdmmc_fb_ck tuned feedback clock selected as receive clock</td>
<td></td>
<td>11</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00:</td>
<td>00: sdmmc_io_in_ck selected as receive clock</td>
<td></td>
<td>01:</td>
<td>SDMMC_CKIN feedback clock selected as receive clock</td>
<td></td>
<td>10:</td>
<td>sdmmc_fb_ck tuned feedback clock selected as receive clock</td>
<td></td>
<td>11:</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01:</td>
<td>SDMMC_CKIN feedback clock selected as receive clock</td>
<td></td>
<td>10:</td>
<td>sdmmc_fb_ck tuned feedback clock selected as receive clock</td>
<td></td>
<td>11:</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td>Bits 21:20</td>
<td>SELCLKRX[1:0]: Receive clock selection</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>00: sdmmc_io_in_ck selected as receive clock</td>
<td></td>
<td>01</td>
<td>SDMMC_CKIN feedback clock selected as receive clock</td>
<td></td>
<td>10</td>
<td>sdmmc_fb_ck tuned feedback clock selected as receive clock</td>
<td></td>
<td>11</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01:</td>
<td>SDMMC_CKIN feedback clock selected as receive clock</td>
<td></td>
<td>10:</td>
<td>sdmmc_fb_ck tuned feedback clock selected as receive clock</td>
<td></td>
<td>11:</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td>Bit 19</td>
<td>BUSSPEED: Bus speed for selection of SDMMC operating modes</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected</td>
<td></td>
<td>01</td>
<td>1: SDR50, DDR50, SDR104, HS200 bus speed mode selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01:</td>
<td>DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected</td>
<td></td>
<td>10</td>
<td>1: SDR50, DDR50, SDR104, HS200 bus speed mode selected</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11:</td>
<td>Reserved (select sdmmc_io_in_ck)</td>
<td></td>
<td>Bit 18</td>
<td>DDR: Data rate signaling</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0: SDR Single data rate signaling</td>
<td></td>
<td>01</td>
<td>1: DDR double data rate signaling</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>DDR double data rate signaling</td>
<td></td>
<td>Bit 17</td>
<td>HWFC_EN: Hardware flow control enable</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00</td>
<td>0: Hardware flow control is disabled</td>
<td></td>
<td>01</td>
<td>1: Hardware flow control is enabled</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>01</td>
<td>Hardware flow control is enabled</td>
<td></td>
<td></td>
<td>When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in Section 59.10.11.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 21:20 SELCLKRX[1:0]: Receive clock selection

- 00: sdmmc_io_in_ck selected as receive clock
- 01: SDMMC_CKIN feedback clock selected as receive clock
- 10: sdmmc_fb_ck tuned feedback clock selected as receive clock
- 11: Reserved (select sdmmc_io_in_ck)

Bit 19 BUSSPEED: Bus speed for selection of SDMMC operating modes

- 0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected
- 1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 DDR: Data rate signaling selection

- 0: SDR Single data rate signaling
- 1: DDR double data rate signaling

Bit 17 HWFC_EN: Hardware flow control enable

- 0: Hardware flow control is disabled
- 1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in Section 59.10.11.
Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:
   - Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
   - SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:
   - Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
   - Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
   - SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:
   - Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:
   - Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
   - Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
   - SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)
01: 4-bit wide bus mode: SDMMC_D[3:0] used
10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled
1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)
0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2
0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4
0x0XX: ..
0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256
0xXXX: ..
0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046
Note: While the SD/SDIO card or eMMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

### 59.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

<table>
<thead>
<tr>
<th>Bits 31:0 CMDARG[31:0]</th>
<th>Command argument</th>
</tr>
</thead>
</table>
| These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

### 59.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).
Bits 31:17  Reserved, must be kept at reset value.

Bit 16  **CMDSusPend**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end
- This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
- CMDSusPend = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.
- CMDSusPend = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15  **BOOT**: Enable boot mode procedure
- 0: Boot mode procedure disabled
- 1: Boot mode procedure enabled

Bit 14  **BOOTMODE**: Select the boot mode procedure to be used
- This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
- 0: Normal boot mode procedure selected
- 1: Alternative boot mode procedure selected

Bit 13  **DTHold**: Hold new data block transmission and reception in the DPSM
- If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12  **CPSMEN**: Command path state machine (CPSM) enable bit
- This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
- If this bit is set, the CPSM is enabled.
- When DTEN = 1, no command is transferred nor boot procedure is started. CPSMEN is cleared to 0.
- During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11  **WAITPend**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM
- This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
- WAITPend is only taken into account when DTMODE = MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10  **WAITINT**: CPSM waits for interrupt request
- If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
- If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8  **WAITRESP[1:0]**: Wait for response bits
- This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
- They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
- 00: No response, expect CMDSENT flag
- 01: Short response, expect CMDREND or CCRCFAIL flag
- 10: Short response, expect CMDREND flag (No CRC)
- 11: Long response, expect CMDREND or CCRCFAIL flag
Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM.

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM.

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

---

**Note:**

1. At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2. MultiMediaCard can send two kinds of responses: short responses, 48 bits, or long responses, 136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

---

### 59.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.
59.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)
Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>CARDSTATUS[31:0]</th>
<th>Card status according to Table below</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>r r r r r r r r r r r r r r r</td>
<td>(See Table 634.)</td>
</tr>
</tbody>
</table>

The card status size is 32 or 128 bits, depending on the response type.

<table>
<thead>
<tr>
<th>Table 634. Response type and SDMMC_RESPxR registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Register(1)</td>
</tr>
<tr>
<td>SDMMC_RESP1R</td>
</tr>
<tr>
<td>SDMMC_RESP2R</td>
</tr>
<tr>
<td>SDMMC_RESP3R</td>
</tr>
<tr>
<td>SDMMC_RESP4R</td>
</tr>
</tbody>
</table>

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

59.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024
Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>DATATIME[31:16]</th>
<th>DATATIME[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Bits 31:0 DATETIME[31:0]: Data and R1b busy timeout period
   This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).
   Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

59.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028
Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>DATALENGTH[24:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATALENGTH[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 DATALENGTH[24:0]: Data length value
   This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).
   Number of data bytes to be transferred.
   When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)
   When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.
59.10.9  SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C
Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>FIFORST: FIFO reset, flushes any remaining data</td>
</tr>
<tr>
<td>29</td>
<td>This bit can only be written by firmware when</td>
</tr>
<tr>
<td>28</td>
<td>IDMAEN= 0 and DPSM is active (DPSMACT = 1).</td>
</tr>
<tr>
<td>27</td>
<td>This bit only takes effect when a transfer error</td>
</tr>
<tr>
<td>26</td>
<td>or transfer hold occurs.</td>
</tr>
<tr>
<td>25</td>
<td>0: FIFO not affected.</td>
</tr>
<tr>
<td>24</td>
<td>1: Flush any remaining data and reset the FIFO</td>
</tr>
<tr>
<td>23</td>
<td>pointers. This bit is automatically cleared to</td>
</tr>
<tr>
<td>22</td>
<td>0 by hardware when DPSM gets inactive (DPSMACT</td>
</tr>
<tr>
<td>21</td>
<td>= 0).</td>
</tr>
<tr>
<td>20</td>
<td>Bit 13</td>
</tr>
<tr>
<td>19</td>
<td>BOOTACKEN: Enable the reception of the boot</td>
</tr>
<tr>
<td>18</td>
<td>acknowledgment</td>
</tr>
<tr>
<td>17</td>
<td>This bit can only be written by firmware when</td>
</tr>
<tr>
<td>16</td>
<td>DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td>15</td>
<td>0: Boot acknowledgment disabled, not expected</td>
</tr>
<tr>
<td>14</td>
<td>to be received</td>
</tr>
<tr>
<td>13</td>
<td>1: Boot acknowledgment enabled, expected to be</td>
</tr>
<tr>
<td>12</td>
<td>received</td>
</tr>
<tr>
<td>11</td>
<td>SDIOEN: SD I/O interrupt enable functions</td>
</tr>
<tr>
<td>10</td>
<td>This bit can only be written by firmware when</td>
</tr>
<tr>
<td>9</td>
<td>DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td>8</td>
<td>If this bit is set, the DPSM enables the SD I/O</td>
</tr>
<tr>
<td>7</td>
<td>card specific interrupt operation.</td>
</tr>
<tr>
<td>6</td>
<td>Bit 10</td>
</tr>
<tr>
<td>5</td>
<td>RWMOD: Read Wait mode</td>
</tr>
<tr>
<td>4</td>
<td>This bit can only be written by firmware when</td>
</tr>
<tr>
<td>3</td>
<td>DPSM is inactive (DPSMACT = 0).</td>
</tr>
<tr>
<td>2</td>
<td>0: Read Wait control using SDMMC_D2</td>
</tr>
<tr>
<td>1</td>
<td>1: Read Wait control stopping SDMMC_CK</td>
</tr>
<tr>
<td>0</td>
<td>Bit 9</td>
</tr>
<tr>
<td></td>
<td>RWSTOP: Read Wait stop</td>
</tr>
<tr>
<td></td>
<td>This bit is written by firmware and auto cleared</td>
</tr>
<tr>
<td></td>
<td>by hardware when the DPSM moves from the R_W</td>
</tr>
<tr>
<td></td>
<td>state to the Wait_R or idle state.</td>
</tr>
<tr>
<td></td>
<td>0: No Read Wait stop</td>
</tr>
<tr>
<td></td>
<td>1: Enable for Read Wait stop when DPSM is in the</td>
</tr>
<tr>
<td></td>
<td>R_W state.</td>
</tr>
<tr>
<td></td>
<td>Bit 8</td>
</tr>
<tr>
<td></td>
<td>RWSTART: Read Wait start</td>
</tr>
<tr>
<td></td>
<td>If this bit is set, Read Wait operation starts.</td>
</tr>
</tbody>
</table>
Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
- 0000: Block length = $2^0$ = 1 byte
- 0001: Block length = $2^1$ = 2 bytes
- 0010: Block length = $2^2$ = 4 bytes
- 0011: Block length = $2^3$ = 8 bytes
- 0100: Block length = $2^4$ = 16 bytes
- 0101: Block length = $2^5$ = 32 bytes
- 0110: Block length = $2^6$ = 64 bytes
- 0111: Block length = $2^7$ = 128 bytes
- 1000: Block length = $2^8$ = 256 bytes
- 1001: Block length = $2^9$ = 512 bytes
- 1010: Block length = $2^{10}$ = 1024 bytes
- 1011: Block length = $2^{11}$ = 2048 bytes
- 1100: Block length = $2^{12}$ = 4096 bytes
- 1101: Block length = $2^{13}$ = 8192 bytes
- 1110: Block length = $2^{14}$ = 16384 bytes
- 1111: Reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transferred data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transferred.)
When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transferred)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- 00: Block data transfer ending on block count.
- 01: SDIO multibyte data transfer.
- 10: eMMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
- 11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
- 0: From host to card.
- 1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.
This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or eMMC cards).
- 0: Do not start data transfer without CPSM data transfer command.
- 1: Start data transfer without CPSM data transfer command.

59.10.10 **SDMMC data counter register (SDMMC_DCNTR)**
Address offset: 0x030
Reset value: 0x0000 0000
This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and
when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:25  Reserved, must be kept at reset value.

Bits 24:0  **DATACOUNT[24:0]**: Data count value
- When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

**Note:** This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transferred.

### 59.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:
- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29  Reserved, must be kept at reset value.

- **Bit 28**  **IDMABTC**: IDMA buffer transfer complete
  - The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

- **Bit 27**  **IDMATE**: IDMA transfer error
  - The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

- **Bit 26**  **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure
  - The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
Bit 25 **VSWEND**: Voltage switch critical timing section completion
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOT**: SDIO interrupt received
   The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
   This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
   0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
   1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
   This bit is reset to not busy when the SDMMC_D0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
   0: card signals not busy on SDMMC_D0.
   1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
   This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
   This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
   This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
   This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
   There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
   At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)
   This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)
   This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
   Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
Bit 10 **DBCKEND**: Data block sent/received  
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state  
or  
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOlD**: Data transfer Hold  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly  
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENt**: Command sent (no response required)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDRENd**: Command response received (CRC check passed, or no CRC)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
  
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)  
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

**Note**:  
FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.
59.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038
Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>IDMA BTCC</td>
<td>IDMA buffer transfer complete clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the IDMABTC flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: IDMABTC not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: IDMABTC cleared</td>
</tr>
<tr>
<td>27</td>
<td>IDMA TEC</td>
<td>IDMA transfer error clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the IDMATE flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: IDMATE not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: IDMATE cleared</td>
</tr>
<tr>
<td>26</td>
<td>CK STOPC</td>
<td>CKSTOP flag clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the CKSTOP flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: CKSTOP not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CKSTOP cleared</td>
</tr>
<tr>
<td>25</td>
<td>VSW ENDC</td>
<td>VSWEND flag clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the VSWEND flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: VSWEND not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: VSWEND cleared</td>
</tr>
<tr>
<td>24</td>
<td>ACK TIMEOUTC</td>
<td>ACKTIMEOUT flag clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the ACKTIMEOUT flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: ACKTIMEOUT not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: ACKTIMEOUT cleared</td>
</tr>
<tr>
<td>23</td>
<td>ACK FAILC</td>
<td>ACKFAIL flag clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the ACKFAIL flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: ACKFAIL not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: ACKFAIL cleared</td>
</tr>
<tr>
<td>22</td>
<td>SDIO ITC</td>
<td>SDIOIT flag clear bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Set by software to clear the SDIOIT flag.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: SDIOIT not cleared</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: SDIOIT cleared</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMA BTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMA TEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CK STOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSW ENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACK TIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACK FAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIO ITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared
Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTE**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREN flag clear bit
Set by software to clear the CMDREN flag.
0: CMDREN not cleared
1: CMDREN cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDR flag clear bit
Set by software to clear the TXUNDR flag.
0: TXUNDR not cleared
1: TXUNDR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared
59.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C
Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>IDMA BTCIE</td>
<td>IDMA buffer transfer complete interrupt enable</td>
</tr>
<tr>
<td>27</td>
<td>CKSTOP IE</td>
<td>Voltage switch clock stopped interrupt enable</td>
</tr>
<tr>
<td>26</td>
<td>VSW ENDIE</td>
<td>Voltage switch critical timing section completion interrupt enable</td>
</tr>
<tr>
<td>25</td>
<td>ACK TIMEOUT IE</td>
<td>Acknowledge timeout interrupt enable</td>
</tr>
<tr>
<td>24</td>
<td>ACK FAILIE</td>
<td>Acknowledge fail flag interrupt enable</td>
</tr>
<tr>
<td>23</td>
<td>SDIO ENDIE</td>
<td>SDIO interrupt enable</td>
</tr>
<tr>
<td>22</td>
<td>BUSY DO ENDIE</td>
<td>Busy do end interrupt</td>
</tr>
<tr>
<td>21</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>TX FIFO IE</td>
<td>Transmit FIFO interrupt</td>
</tr>
<tr>
<td>18</td>
<td>RX FIFO IE</td>
<td>Receive FIFO interrupt</td>
</tr>
<tr>
<td>17</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>RX FIFO HFIE</td>
<td>Receive FIFO high fairness interrupt</td>
</tr>
<tr>
<td>14</td>
<td>TX FIFO HFIE</td>
<td>Transmit FIFO high fairness interrupt</td>
</tr>
<tr>
<td>13</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>12</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>11</td>
<td>DA BORT IE</td>
<td>Data abort interrupt</td>
</tr>
<tr>
<td>10</td>
<td>DBCK ENDIE</td>
<td>Data back up clock end interrupt</td>
</tr>
<tr>
<td>9</td>
<td>DHOLD IE</td>
<td>Data hold interrupt</td>
</tr>
<tr>
<td>8</td>
<td>DATA ENDIE</td>
<td>Data end interrupt</td>
</tr>
<tr>
<td>7</td>
<td>CMD SENT IE</td>
<td>Command sent interrupt</td>
</tr>
<tr>
<td>6</td>
<td>CMDR ENDIE</td>
<td>Command reply end interrupt</td>
</tr>
<tr>
<td>5</td>
<td>RX OVER RIE</td>
<td>Receive over run interrupt</td>
</tr>
<tr>
<td>4</td>
<td>TX UNDER RIE</td>
<td>Transmit under run interrupt</td>
</tr>
<tr>
<td>3</td>
<td>D TIME OUTIE</td>
<td>Data time out interrupt</td>
</tr>
<tr>
<td>2</td>
<td>C TIME OUTIE</td>
<td>Command time out interrupt</td>
</tr>
<tr>
<td>1</td>
<td>DCRC FAILIE</td>
<td>DCRC fail interrupt</td>
</tr>
<tr>
<td>0</td>
<td>CCRC FAILIE</td>
<td>CCRC fail interrupt</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
- Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
  0: IDMA buffer transfer complete interrupt disabled
  1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOP IE**: Voltage switch clock stopped interrupt enable
- Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
  0: Voltage switch clock stopped interrupt disabled
  1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
- Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
  0: Voltage switch critical timing section completion interrupt disabled
  1: Voltage switch critical timing section completion interrupt enabled
Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled
Bit 9 **D HOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **D ATE N D I E**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSEN T I E**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CM DREN D I E**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: Command response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUN DERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **D TIM EOUT I E**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CT IM EOUT I E**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRFFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRFFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled
59.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040
Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

59.10.15 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050
Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.
59.10.16 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

| Bits 31:17 | Reserved, must be kept at reset value. |
| Bits 16:5  | **IDMABNDT[11:0]**: Number of bytes per buffer |
| Bits 4:0   | Reserved, must be kept at reset value. |

**IDMABNDT[11:0]**

- This 12-bit value must be multiplied by 8 to get the size of the buffer in 32-bit words and by 32 to get the size of the buffer in bytes.
- Example: IDMABNDT = 0x001: buffer size = 8 words = 32 bytes.
- Example: IDMABNDT = 0x800: buffer size = 16384 words = 64 Kbyte.
- These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).
59.10.17 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058
Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IDMABASE[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>IDMABASE[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  rw  r  r</td>
</tr>
</tbody>
</table>

Bits 31:0 IDMABASE[31:0]: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

59.10.18 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>IDMALAR[13:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw r r</td>
</tr>
</tbody>
</table>

Bits 31:0 IDMALAR[31:0]: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).
Bit 31  **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode  
(SDMMC_IDMACTRLR.IDMABMODE select linked list mode)  
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).  
0: SDMMC_IDMALAR is not to be updated, last linked list item.  
1: SDMMC_IDMALAR is to be updated from linked list table.  

Bit 30  **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode  
(SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)  
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).  
0: SDMMC_IDMABSIZE is not to be updated from next linked list table.  
1: SDMMC_IDMABSIZE is to be updated from next linked list table.  

Bit 29  **ABR**: Acknowledge linked list buffer ready  
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).  
This bit is not taken into account when starting the first linked list buffer from the software  
programmed register information. ABR is only taken into account on subsequent loaded  
linked list items.  
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be  
generated).  
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transferred by  
IDMA.  

Bits 28:16  Reserved, must be kept at reset value.  

Bits 15:2  **IDMALA[13:0]**: Word aligned linked list item address offset  
Linked list item offset pointer to the base of the next linked list item structure.  
Linked list item base address is IDMABA + IDMALA.  
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).  

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

59.10.19  **SDMMC IDMA linked list memory base register**  
(SDMMC_IDMABAR)  
Address offset: 0x068  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>IDMABA[29:14]</td>
<td>rw</td>
</tr>
<tr>
<td>30</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>22</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>20</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>18</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>16</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Value</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>rw</td>
<td>IDMABA[13:0]</td>
<td>rw</td>
</tr>
<tr>
<td>14</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>12</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>10</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>8</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>6</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>4</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>2</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>0</td>
<td>rw</td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:2  **IDMABA[29:0]**: Word aligned Linked list memory base address  
Linked list memory base pointer.  
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).  

Bits 1:0  Reserved, must be kept at reset value.
Secure digital input/output MultiMediaCard interface (SDMMC)  RM0477

59.10.20  SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)
Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x080 + 0x004 * x</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x =0 to 15)</td>
<td></td>
</tr>
</tbody>
</table>

Reset value: 0x0000 0000

The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.
### Table 635. SDMMC register map (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bit name</th>
<th>Bit range</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010</td>
<td>SDMMC_RESPCMDR</td>
<td>RESPCMD[5:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x014</td>
<td>SDMMC_RESP1R</td>
<td>CARDSTATUS[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x018</td>
<td>SDMMC_RESP2R</td>
<td>CARDSTATUS[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x01C</td>
<td>SDMMC_RESP3R</td>
<td>CARDSTATUS[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x020</td>
<td>SDMMC_RESP4R</td>
<td>CARDSTATUS[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x024</td>
<td>SDMMC_DTIMER</td>
<td>DATETIME[31:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x028</td>
<td>SDMMC_DLENR</td>
<td>DATALENGTH[24:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x02C</td>
<td>SDMMC_DCTRLR</td>
<td>FIFORST</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x030</td>
<td>SDMMC_DCNTR</td>
<td>DATAACCOUNT[24:0]</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x034</td>
<td>SDMMC_STAR</td>
<td>IDMAC Tec</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x038</td>
<td>SDMMC_ICR</td>
<td>IDMAC Tec</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0x03C</td>
<td>SDMMC_MASKR</td>
<td>FIFORST</td>
<td>0x00000000</td>
<td>0x00000000</td>
<td>0x00000000</td>
</tr>
</tbody>
</table>
Table 635. SDMMC register map (continued)

| Offset  | Register name | Name                  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------|-----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x040   | SDMMC_ACKTIMER| ACKTIME[24:0]         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x044   |               |                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x044   |               | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04C   |               |                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x050   | SDMMC_IDMACTRLR| IDMABNDT[11:0]      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x054   | SDMMC_IDMABSZER| IDMABNDT[11:0]      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x058   | SDMMC_IDMABSE | IDMABNDT[11:0]      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x05C   |               |                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x064   | SDMMC_IDMALAR | IDMAL[13:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x068   | SDMMC_IDMABAR | IDMAB[28:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x06C   |               |                       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x080   | SDMMC_FIFORx  | FIFODATA[31:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |               | Reset value           | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Refer to Section 2.3 on page 149 for the register boundary addresses.
60 FD controller area network (FDCAN)

60.1 Introduction

The controller area network (CAN) subsystem (see Figure 883) consists of one CAN module, a shared message RAM, and a configuration block. Refer to the memory map for the base address of each of these parts.

The modules (FDCAN) are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

A 0.8-Kbyte message RAM per FDCAN instance implements filters, receive FIFOs, transmit event FIFOs, and transmit FIFOs.

The CAN subsystem I/O signals and pins are detailed, respectively, in Table 636 and Figure 883.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdcan_ker_ck</td>
<td>Digital input</td>
<td>CAN subsystem kernel clock input</td>
</tr>
<tr>
<td>fdcan_pclk</td>
<td></td>
<td>CAN subsystem APB interface clock input</td>
</tr>
<tr>
<td>fdcan_intr0_it</td>
<td>Digital output</td>
<td>FDCAN interrupt0</td>
</tr>
<tr>
<td>fdcan_intr1_it</td>
<td></td>
<td>FDCAN interrupt1</td>
</tr>
<tr>
<td>fdcan_ts[0:15]</td>
<td>-</td>
<td>External timestamp vector</td>
</tr>
<tr>
<td>FDCAN_RX</td>
<td>Digital input</td>
<td>FDCAN receive pin</td>
</tr>
<tr>
<td>FDCAN_TX</td>
<td>Digital output</td>
<td>FDCAN transmit pin</td>
</tr>
<tr>
<td>APB interface</td>
<td>Digital input/output</td>
<td>Single APP with multiple psel for configuration, control and RAM access</td>
</tr>
</tbody>
</table>
60.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015
- CAN FD with maximum 64 data bytes supported
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two receive FIFOs of three payloads each (up to 64 bytes per payload)
- Separate signaling on reception of high priority messages
- Configurable transmit FIFO/queue of three payloads (up to 64 bytes per payload)
- Transmit event FIFO
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support
60.3 FDCAN functional description

Figure 884. FDCAN block diagram

Dual interrupt lines
The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it.
By programming the EINT0 and EINT1 bits of the FDCAN_ILE register, the interrupt lines can be independently enabled or disabled.

CAN core
The CAN core contains the protocol controller and receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions and supports both 11-bit and 29-bit identifiers.

Sync
This block synchronizes signals from the APB clock domain to the CAN kernel clock domain and vice versa.
Tx handler

The Tx handler controls the message transfer from the message RAM to the CAN core. A maximum of three Tx buffers is available for transmission. The Tx buffer can be used as Tx FIFO or as a Tx queue. Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

Rx handler

The Rx handler controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, for storage of all messages that have passed acceptance filtering. An Rx timestamp is stored together with each message. Up to 28 filters can be defined for 11-bit IDs; up to eight filters for 29-bit IDs.

APB interface

The APB interface connects the FDCAN to the APB bus for configuration registers, controller configuration, and RAM access.

Message RAM interface

The message RAM interface connects the FDCAN access to an external 1-Kbyte message RAM through a RAM controller/arbiter.

60.3.1 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges.

As shown in Figure 885, its operation can be explained simply by splitting the bit time in three segments, as follows:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, having a fixed length of one time quantum (1 × tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard. Its duration is programmable from 1 to 16 time quanta, but can be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but can also be automatically shortened to compensate for negative phase drifts.
The baud rate is the inverse of the bit time (baud rate = 1 / bit time), which, in turn, is the sum of three components (see Figure 885):

\[
\text{bit time} = t_{\text{SyncSeg}} + t_{BS1} + t_{BS2}
\]

Where:
- For the nominal bit time
  \[
  t_q = (\text{NBRP}[8:0] + 1) \times t_{\text{fdcan_tq_clk}}
  \]
  \[
  t_{\text{SyncSeg}} = t_q
  \]
  \[
  t_{BS1} = t_q \times (\text{NTSEG1}[7:0] + 1)
  \]
  \[
  t_{BS2} = t_q \times (\text{NTSEG2}[6:0] + 1)
  \]
  Where NBRP[8:0], NTSEG1[7:0], and NTSEG2[6:0] bitfields belong to the FDCAN_NBTP register.
- For the data bit time
  \[
  t_q = (\text{DBRP}[4:0] + 1) \times t_{\text{fdcan_tq_clk}}
  \]
  \[
  t_{\text{SyncSeg}} = t_q
  \]
  \[
  t_{BS1} = t_q \times (\text{DTSEG1}[4:0] + 1)
  \]
  \[
  t_{BS2} = t_q \times (\text{DTSEG2}[3:0] + 1)
  \]
  Where DBRP[4:0], DTSEG1[4:0], and DTSEG2[3:0] belong to the FDCAN_DBTP register.

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW, so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW, so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. The FDCAN_DBTP and FDCAN_NBTP registers (dedicated, respectively, to data and nominal bit timing) are only accessible when the CCE and INIT of the FDCA_CCCR register are set.

Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.
### 60.3.2 Operating modes

#### Configuration

Access to IP version, hardware, and input clock divider configuration. When the clock divider is set to 0, the primary input clock is used as it is.

#### Software initialization

Software initialization is started by setting the INIT bit of the FDCAN_CCCR register, by software, by a hardware reset, or by entering bus-off state. While the INIT bit is set, message transfers from and to the CAN bus are stopped, and the status of the CAN bus output FDCAN_TX is recessive (high). The EML (error management logic) counters are unchanged. Setting the INIT bit does not change any configuration register. Clearing INIT bit of FDCAN_CCCR finishes the software initialization. Afterwards the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (bus-idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when the INIT bit and the CCE bit of the FDCAN_CCCR register are both set.

The CCE bit of the FDCAN_CCCR register can only be set/cleared while the INIT bit of FDCAN_CCCR is set. The CCE bit is automatically cleared when the INIT bit is cleared.

The following registers are reset when the CCE bit of the FDCAN_CCCR register is set:

- FDCAN_HPMS: High priority message status
- FDCAN_RXF0S: Rx FIFO 0 status
- FDCAN_RXF1S: Rx FIFO 1 status
- FDCAN_TXFQS: Tx FIFO/queue status
- FDCAN_TXBRP: Tx buffer request pending
- FDCAN_TXBTO: Tx buffer transmission occurred
- FDCAN_TXBCF: Tx buffer cancellation finished
- FDCAN_TXEFS: Tx event FIFO status

The timeout counter value (TOC[15:0] bit of the FDCAN_TOCV register) is preset to the value configured by the TOP[15:0] of the FDCAN_TOCC register when the CCE bit of the FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit is set.

The following registers can be written only when the CCE bit is cleared:

- FDCAN_TXBAR: Tx buffer add request
- FDCAN_TXBCR: Tx buffer cancellation request

The TEST and the MON bits of the FDCAN_CCCR register can be set only by software while the INIT and the CCE bits of the FDCAN_CCCR register are both set. Both bits can be reset at any time. The DAR bit of FDCAN_CCCR can only be set/cleared while the INIT and CCE bits are both set.
Normal operation

The FDCAN default operating mode after hardware reset is event-driven CAN communication. TT operation mode is not supported.

Once the FDCAN is initialized and the INIT bit of the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted, the Tx FIFO or the Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the FDCAN protocol:

1. Long frame mode (LFM), where the data field of a CAN frame may be longer than eight bytes
2. Fast frame mode (FFM), where the control field, data field, and CRC field of a CAN frame are transmitted with a higher bit rate compared to the beginning and to the end of the frame

The fast frame mode can be used in combination with the long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are decoded as FDF bit: FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame.

In a CAN FD frame, the two bits following FDF (res and BRS) decide whether the bit rate inside this CAN FD frame is switched. A CAN FD bit rate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for future expansion of the protocol. In case the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting the PXE bit of the FDCAN_PSR register. When protocol exception handling is enabled (PXHD = 0 in FDCAN_CCCR), this causes the operation state to change from receiver (ACT[1:0] = 10 in FDCAN_PSR) to integrating (ACT[1:0] = 00 in FDCAN_PSR) at the next sample point. If protocol exception handling is disabled (PXHD = 1 in FDCAN_CCCR), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming the FDOE bit of the FDCAN_CCCR register. In case FDOE = 1, transmission and reception of CAN FD frames are enabled. Transmission and reception of classic CAN frames are always possible. Whether a CAN FD frame or a classic CAN frame is transmitted can be configured via the FDF bit in the respective Tx buffer element. With FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted even if the FDF bit of a Tx buffer element is set. The FDOE and BRSE bits of the FDCAN_CCCR register can only be changed while the INIT and CCE bits are both set.

With FDOE = 0, the setting of the FDF and BRS bits is ignored, and frames are transmitted in classic CAN format. With FDOE = 1 and BRSE = 0, only the FDF bit of a Tx buffer element is evaluated. With FDOE = 1 and BRSE = 1, transmission of CAN FD frames with bit rate switching is enabled. All Tx buffer elements with FDF and BRSE bits set are transmitted in CAN FD format with bit rate switching.
A mode change during CAN operation is recommended only under the following conditions:

- The failure rate in the CAN FD data phase is significantly higher than in the CAN FD arbitration phase. In this case, disable the CAN FD bit rate switching option for transmissions.
- During system startup, all nodes transmit classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking have to be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non-CAN FD nodes are held in silent mode until programming is complete. Then all nodes switch back to classic CAN communication.

In the FDCAN format, the coding of the DLC differs from that of the standard CAN format. The DLC codes 0 to 8 have the same coding as in standard CAN, the codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to Table 637.

<table>
<thead>
<tr>
<th>DLC</th>
<th>Number of data bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>48</td>
</tr>
<tr>
<td>15</td>
<td>64</td>
</tr>
</tbody>
</table>

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bit rate switch) bit, if this bit is recessive. Before the BRS bit, in the FDCAN arbitration phase, the standard CAN bit timing is used as defined by the FDCAN_DBTP register. In the following FDCAN data phase, the fast CAN bit timing is used as defined by the FDCAN_DBTP register. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bit rate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (t_q), the bit rate in the data phase is 5 Mbit/s.

In both data frame formats (CAN FD long frames and CAN FD fast frames), the value of bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames, the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for FDCAN transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives a FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of an FDCAN transmission, only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX, the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop...
delay. If this delay is greater than TSEG1 (time segment before sample point), and a bit error is detected. Without transceiver delay compensation, the bit rate in the data phase of an FDCAN frame is limited by the transceiver loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bit rates during the FDCAN data phase independent of the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the secondary sample point (SSP). If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay. This is enabled by setting the TDC bit of the FDCAN_DBTP register, and described in detail in the ISO11898-1 specification.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by TDCO[6:0] of FDCAN_TDCR. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example, half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of mtq (minimum time quantum, one period of fdcan_to_ck clock).

The TDCV[6:0] bitfield of the FDCAN_PSR register shows the actual transmitter delay compensation value. TDCV[6:0] is cleared when the INIT is set in the FDCAN_CCCR. It is updated at each transmission of an FD frame while the TDC bit of the FDCAN_DBTP register is set.

The following boundary conditions have to be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCAN_Tx to FDCAN_Rx and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than 6-bit times in the data phase.
- The sum of the measured delay from FDCAN_TX to FDCAN_RX and the configured transmitter delay compensation offset TDCO[6:0] has to be lower than or equal to $127 \times mtq$. If the sum exceeds this value, the maximum value ($127 \times mtq$) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs.

If transmitter delay compensation is enabled by setting the TDC bit of the FDCAN_DBTP; the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is one mtq.
To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming the TDCF[6:0] bitfield of the FDCAN_TDCR register. This defines a minimum value for the SSP position. Dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least TDCF[6:0] and FDCAN_RX is low.

**Restricted operation mode**

In restricted operation mode, the node is able to receive data and remote frames, and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits. Instead, it waits for the occurrence of a bus-idle condition to resynchronize itself to the CAN communication. The error counters (REC[6:0] and TEC[7:0] in FDCAN_ECR) are frozen while the error logging (CEL[7:0]) is active. The software can set the FDCAN into restricted operation mode by setting the ASM bit of FDCAN_CCCR. The bit can only be set by software when both CCE and INIT bits are set in FDCAN_CCCR. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler is not able to read data from the message RAM in time. To leave restricted operation mode, the software has to clear the ASM bit of FDCAN_CCCR.

The restricted operation mode can be used in applications that adapt themselves to different CAN bit rates. In this case, the application tests different bit rates and leaves the restricted operation mode after it has received a valid frame.

**Note:** The restricted operation mode must not be combined with the loop-back mode (internal or external).
Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting the MON bit of the FDCAN_CCCR register. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus. If the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN can monitor it, even if the CAN bus remains in recessive state. In bus monitoring mode, the FDCAN_TXBRP register is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. Figure 887 shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Disabled automatic retransmission mode (DAR)

According to the CAN specification (see ISO11898-1, 6.3.3 Recovery Management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled. The DAR mode can be disabled through the DAR bit of the FDCAN_CCCR register.

Frame transmission in DAR mode

In DAR mode, all transmissions are automatically canceled after they have been started on the CAN bus. A Tx buffer Tx request pending bit (TRPx in FDCAN_TXBRP) is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, when it has been aborted due to lost arbitration, or when an error has occurred during frame transmission.

- Successful transmission
  - The corresponding Tx buffer transmission occurred bit TOx is set in FDCAN_TXBTO register.
  - The corresponding Tx buffer cancellation finished bit CFx is cleared in the FDCAN_TXBCF register.
• Successful transmission in spite of cancellation
  – The corresponding Tx buffer transmission occurred bit TOx is set in the FDCAN_TXBTO register.
  – The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

• Arbitration loss or frame transmission disturbed
  – The corresponding Tx buffer transmission occurred bit TOx is cleared in the FDCAN_TXBTO register.
  – The corresponding Tx buffer cancellation finished bit CFx is set in the FDCAN_TXBCF register.

In case of a successful frame transmission, and if the storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

**Power-down (Sleep mode)**

### Power-down entry

The FDCAN can be set into power-down mode controlled by setting the CSR bit of the FDCAN_CCCR register. As long as the clock stop request is active, CSR is read as 1.

When all pending transmission requests have completed, the FDCAN waits until the bus-idle state is detected. The FDCAN then sets the INIT bit of the FDCAN_CCCR register to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power-down by setting the CSA bit of the FDCAN_CCCR register. In this state, before the clocks are switched off, further register accesses can be made. A write access to the INIT bit has no effect. Now, the module clock inputs can be switched off.

### Power-down exit

To leave power-down mode, the application has to turn on the module clocks before clearing the CSR bit. The FDCAN acknowledges this by clearing the CSA bit. Afterwards, the application can restart CAN communication by clearing the INIT bit.

**Test modes**

To enable write access to **FDCAN test register (FDCAN_TEST)**, the TEST bit of the FDCAN_CCCR register must be set, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming the TX[1:0] bitfield of the FDCAN_TEST register. In addition to its default function (the serial data output), it can drive the CAN sample point signal to monitor the FDCAN bit timing as well as drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from the RX bit of FDCAN_TEST. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there can be a delay of several APB clock periods between writing to TX[1:0] until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via RX.

*Note:* Test modes must be used for production tests or self-test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.
External loop-back mode

The FDCAN can be set in external loop-back mode by setting the LBCK bit of the FDCAN_TEST register. In loop-back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. Figure 888 shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop-back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop-back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop-back mode

Internal loop-back mode is entered by setting both the LBCK bit of FDCAN_TEST and the MON bit of FDCAN_CCR. This mode can be used for a “hot self-test”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, FDCAN_RX pin is disconnected from the FDCAN and FDCAN_TX pin is held recessive. Figure 888 shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop-back mode.

Timestamp generation

For timestamp generation, the FDCAN supplies a 16-bit wrap-around counter. A prescaler (TCP[3:0] of FDCAN_TSCC) can be configured to clock the counter in multiples of CAN bit times (1 to 16). The counter is readable via the TCV[15:0] bitfield of the FDCAN_TSCV register. A write access to TSCV15:0 resets the counter to 0. When the timestamp counter wraps around, the interrupt flag (TSW bit of FDCAN_ISR) is set.

On start of frame reception/transmission, the counter value is captured and stored into the timestamp section of an Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.
By programming TSS[1:0] of FDCAN_TSCC, a 16-bit timestamp can be used.

**Debug mode behavior**

In debug mode, the set/reset on read feature is automatically disabled during the debugger register access, and enabled during normal MCU operation.

**Timeout counter**

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as a down-counter and uses the same prescaler controlled by TCP[3:0] of FDCAN_TSCC as the timestamp counter. The timeout counter is configured via the FDCAN_TOCC register. The actual counter value can be read from the TOC[15:0] bitfield of FDCAN_TOCV. The timeout counter can only be started while the INIT bit of FDCAN_CCCR is cleared. It is stopped when INIT is set, for example, when the FDCAN enters bus-off state.

The operation mode is selected by TOS[1:0] of FDCAN_TOCC. When operating in continuous mode, the counter starts when INIT is cleared. A write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the TOO interrupt flag is set in the FDCAN_IR register. In continuous mode, the counter is immediately restarted at TOP[15:0].

*Note:* *The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented may vary due to the synchronization/resynchronization mechanism of the CAN core. If the baud rate switch feature in FDCAN is used, the timeout counter is clocked differently in the arbitration and data fields.*
60.3.3 Message RAM

The message RAM is 32-bit wide, and the FDCAN module is configured to allocate up to 212 words in it. It is not necessary to configure each of the sections shown in Figure 889.

Figure 889. Message RAM configuration

When the FDCAN addresses the message RAM, it addresses 32-bit words (aligned), not a single byte. The RAM addresses are 32-bit words, that is, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

In case of multiple instances, the RAM start address for the FDCANn is computed by end address + 4 of FDCANn - 1, and the FDCANn end address is computed by FDCANn start address + 0x0350 - 4.

As an example, for two instances:
- FDCAN1:
  - Start address 0x0000
  - End address 0x034C (as in Figure 889)
- FDCAN2:
  - Start address = 0x034C (FDCAN1 end address) + 4 = 0x0350
  - End address = 0x0350 (FDCAN2 start address) + 0x0350 - 4 = 0x069C.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to one of the two Rx FIFOs, as well as the Rx FIFO put and get indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and another for extended identifiers. These filters can be assigned to Rx FIFO 0 or Rx FIFO 1. For acceptance filtering, each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element, and the following filter elements are not evaluated for this message.
The main features are:

- Each filter element can be configured as
  - Range filter (from - to)
  - Filter for one or two dedicated IDs
  - Classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering.
- Each filter element can be enabled/disabled individually.
- Filters are checked sequentially, execution stops with the first matching filter element.

Related configuration registers are:

- Global filter configuration (RXGFC)
- Extended ID AND mask (XIDAM)

Depending on the configuration of the filter element (SFEC[2:0]/EFEC[2:0]), a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Reject received frame
- Set the high priority message interrupt flag HPM in FDCAN_IR
- Set the high priority message interrupt flag HPM in FDCAN_IR, and store the received frame in FIFO 0 or FIFO 1.

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx FIFO. If the CAN protocol controller has detected an error condition (for example, CRC error), this message is discarded with the following impact:

- **Rx FIFO**
  The put index of the matching Rx FIFO is not updated, but the related Rx FIFO element is partly overwritten with the received data. For error type, see LEC[2:0] and DLEC[2:0] bitfields of the FDCAN_PSR register. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in **Rx FIFO overwrite mode** have to be considered.

**Note:** *When an accepted message is written to one of the two Rx FIFOs, the unmodified received identifier is stored independently from the used filters. The result of the acceptance filter process strongly depends on the sequence of configured filter elements.*

**Range filter**

The filter matches for all received frames with message IDs in the range defined by SF1ID/SF2ID and EF1ID/EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- **EFT[1:0] = 00**: the message ID of received frames is AND-ed with the extended ID AND mask (XIDAM) before the range filter is applied.
- **EFT[1:0] = 11**: the extended ID AND mask (XIDAM) is not used for range filtering.
Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element has to be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

The classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID/EF1ID is used as message ID filter, while SF2ID/EF2ID is used as filter mask.

0 bit at the filter mask masks out the corresponding bit position of the configured ID filter. For example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only the bits of the received message ID where the corresponding mask bits are 1 are relevant for acceptance filtering.

In case all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 890 shows the flow for standard message ID (11-bit identifier) filtering. The standard message ID filter element is described in Section 60.3.8.

The standard message filtering is controlled by the FDCAN_RXGFC register. The standard message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.
Figure 890. Standard message ID filter path

- Valid frame received
  - 11-bit Bit identifier
  - Remote frame
    - Yes: Reject remote frame
    - No: Receive filter list enabled
      - RXGFC[RRFS] = 0
      - RXGFC[LSS[7:0]] > 0
        - Yes: Accept non-matching frames
          - RXGFC[ANFS[1]] = 1
            - RXGFC[ANFS[1]] = 0
              - Target FIFO full
                - Yes: Discard frame
                - No: Append to target FIFO
      - RXGFC[LSS[7:0]] = 0
        - Yes: Accept non-matching frames
          - RXGFC[ANFS[1]] = 1
            - RXGFC[ANFS[1]] = 0
              - Target FIFO full
                - Yes: Discard frame
                - No: Append to target FIFO
Extended message ID filtering

Figure 891 shows the flow for extended message ID (29-bit identifier) filtering. The extended message ID filter element is described in Section 60.3.9.

The extended message filtering is controlled by the FDCAN_RXGFC register. The extended message ID, the remote transmission request bit (RTR), and the identifier extension bit (IDE) of the received frames are compared against the list of configured filter elements.

Figure 891. Extended message ID filter path

The extended ID AND mask (XIDAM) is AND-ed with the received identifier before the filter list is executed.
Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can hold up to three elements each.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see Acceptance filter. The Rx FIFO element is described in Section 60.3.5.

When an Rx FIFO full condition is signaled by RFnF in FDCAN_IR (where n is the FIFO number), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the interrupt flag RFnL is set in the FDCAN_IR register.

When reading from an Rx FIFO, the Rx FIFO get index (FnGl of FDCAN_RXFnS) + FIFO element size has to be added to the corresponding Rx FIFO start address (FnSA).

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by clearing the FnOM bit in the FDCAN_RXGFC register. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (FnPI = FnGI in FDCAN_RXFnS), no further messages are written to the corresponding Rx FIFO until at least one message has been read out and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by FnF = 1 in FDCAN_RXFnS. In addition, the RFnF interrupt flag is set in FDCAN_IR.

In case a message is received while the corresponding Rx FIFO is full, this message is discarded, and the message lost condition is signaled by setting RFnL bit in FDCAN_RXFnS. In addition, the RFnL interrupt flag is set in FDCAN_IR.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by setting the FnOM bit of the FDCAN_RXGFC register.

When an Rx FIFO full condition (FnPI = FnGl of FDCAN_RXFnS) is signaled by FnF = 1 in FDCAN_RXFnS, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get indices are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading from the Rx FIFO elements must start at least at get index + 1. This is because it may happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case, inconsistent data can be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO.

After reading from the Rx FIFO, the number of the last element read has to be written to the Rx FIFO acknowledge index (FnA of FDCAN_RXFnA). This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (FnF = 0 in FDCAN_RXFnS).
Tx handling

The Tx handler handles transmission requests for the Tx FIFO and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get indices, and the Tx event FIFO. Up to three Tx buffers can be set up for message transmission. The CAN message data field is configured to 64 bytes. The Tx FIFO allocates eighteen 32-bit words for storage of a Tx element.

Table 638. Possible configurations for frame transmission

<table>
<thead>
<tr>
<th>CCCR</th>
<th>FDOE</th>
<th>FDF</th>
<th>BRS</th>
<th>Frame transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ignored</td>
<td>0</td>
<td>Ignored</td>
<td>Ignored</td>
<td>Classic CAN</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Ignored</td>
<td>Classic CAN</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Ignored</td>
<td>FD without bit rate switching</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Ignored</td>
<td>Classic CAN</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>FD without bit rate switching</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>FD with bit rate switching</td>
</tr>
</tbody>
</table>

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register (FDCAN_TXBRP) is updated, or when a transmission has been started.

Transmit pause

The transmit pause feature is intended for use in CAN systems where the CAN message identifiers are permanently specified to specific values and cannot easily be changed. These message identifiers can have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

As an example, if CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, it waits, after the first successful message transmission, for two CAN bit times of bus-idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, these messages are started in the idle time, and they would not need to arbitrate with the next message of ECU-1. After having received a message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit of the CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. By default, this feature is disabled (TXP = 0 in FDCAN_CCCR).
This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

**Tx FIFO**

Tx FIFO operation is configured by clearing the TFQM bit of the FDCAN_TXBC register. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index (TFGI[1:0] bitfield of FDCAN_TXFQS). After each transmission, the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order that these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level (TFFL[2:0] bitfield of FDCAN_TXFQS) as the difference between the get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages have to be written to the Tx FIFO starting with the Tx buffer referenced by the put index (TFQPI[1:0] bitfield of FDCAN_TXFQS). An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (TFQF = 1 in FDCAN_TXFQS) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by setting the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via the FDCA_TXBAR register. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with a transmission request is pending and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO Free Level remain unchanged.

A Tx FIFO element allocates eighteen 32-bit words in the message RAM. The Therefore, the start address of the next available (free) Tx FIFO buffer, is calculated by adding four times the put index TFQPI[1:0] (0 … 2) to the Tx buffer start address TBSA.

**Tx queue**

Tx queue operation is configured by setting the TFQM of the FDCAN_TXBC register. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority).

In case of mixing of standard and extended message IDs, the standard message IDs are compared to bits [28:18] of extended message IDs.

In case multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages have to be written to the Tx buffer referenced by the put index (TFQPI[1:0] in FDCAN_TXFQS). An add request cyclically increments the put index to the next free Tx buffer. In case the Tx queue is full (TFQF = 1 in FDCAN_TXFQS), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.
The application can use the FDCAN_TXBRP register instead of the put index and can place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates eighteen 32-bit words in the message RAM. The start address of Therefore, the next available (free) Tx queue buffer is calculated by adding four times the Tx queue put index TFQPI[1:0] (0 ... 2) to the Tx buffer start address TBSA.

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a Tx queue buffer, the host has to write 1 to the corresponding bit position (= number of Tx buffer) of the FDCAN_TXBCR register. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of the FDCAN_TXBCF register.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN_TXBRP bit remains set as long as the transmission is in progress. If the transmission is successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission is not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: In case a pending transmission is canceled immediately before it has been started, there is a short time window where no transmission is started even if another message is pending in the node. This can enable another node to transmit a message that can have a priority lower than that of the second message in the node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO is configured to three elements. The Tx event FIFO element is described in Tx FIFO.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling that is, a Tx buffer holds only the message to be transmitted, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by the TEFF bit of the FDCAN_IR, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the TEFL interrupt flag is set in the FDCAN_IR register.

When reading from the Tx event FIFO, the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) has to be added twice to the Tx event FIFO start address EFSA.
60.3.4 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index (see Section 60.4.23 and Section 60.4.25). Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value = index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care has to be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case, the FIFO acknowledge index must not be written because this would set the get index to a wrong position and alter the FIFO fill level. In this case, some of the older FIFO elements would be lost.

Note: The application has to ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

60.3.5 FDCAN Rx FIFO element

Two Rx FIFOs are configured in the message RAM. Each Rx FIFO section can be configured to store up to three received messages. The structure of an Rx FIFO element is described in Table 639. The description is provided in Table 640.

Table 639. Rx FIFO element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>ANMF</td>
<td>FIDX[6:0]</td>
<td>Res.</td>
<td>FDF</td>
<td>BRS</td>
<td>DLC[3:0]</td>
<td>RXTS[15:0]</td>
<td></td>
</tr>
<tr>
<td>R2</td>
<td>DB3[7:0]</td>
<td>DB2[7:0]</td>
<td>DB1[7:0]</td>
<td>D[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R3</td>
<td>DB7[7:0]</td>
<td>DB6[7:0]</td>
<td>DB5[7:0]</td>
<td>DB4[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rn</td>
<td>DBm[7:0]</td>
<td>DBm-1[7:0]</td>
<td>DBm-2[7:0]</td>
<td>DBm-3[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The element size configured for storage of CAN FD messages is set to 64-byte data field.

Table 640. Rx FIFO element description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| R0 Bit 31 ESI | Error state indicator
  – 0: Transmitting node is error active
  – 1: Transmitting node is error passive |
| R0 Bit 30 XTD | Extended identifier
  Signals to the host whether the received frame has a standard or extended identifier.
  – 0: 11-bit standard identifier
  – 1: 29-bit extended identifier |
**Table 640. Rx FIFO element description (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| R0 Bit 29 RTR | Remote transmission request  
Signals to the host whether the received frame is a data frame or a remote frame.  
– 0: Received frame is a data frame  
– 1: Received frame is a remote frame |
| R0 Bits 28:0 ID[28:0] | Identifier  
Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18]. |
| R1 Bit 31 ANMF | Accepted non-matching frame  
Acceptance of non-matching frames can be enabled via ANFS[1:0] and ANFE[1:0] bitfield of FDCAN_RXGFC.  
– 0: Received frame matching filter index FIDX  
– 1: Received frame did not match any Rx filter element |
| R1 Bits 30:24 FIDX[6:0] | Filter index  
0-27=Index of matching Rx acceptance filter element (invalid if ANMF = 1).  
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC. |
| R1 Bit 21 FDF | FD format  
– 0: Standard frame format  
– 1: FDCAN frame format (new DLC-coding and CRC) |
| R1 Bit 20 BRS | Bit rate switch  
– 0: Frame received without bit rate switching  
– 1: Frame received with bit rate switching |
| R1 Bits 19:16 DLC[3:0] | Data length code  
– 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes  
– 9-15: Classic CAN: received frame has 8 data bytes  
– 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes |
| R1 Bits 15:0 RXTS[15:0] | Rx timestamp  
Timestamp Counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler TCP[3:0] of FDCAN_TSCC. |
| R2 Bits 31:24 DB3[7:0] | Data byte 3 |
| R2 Bits 23:16 DB2[7:0] | Data byte 2 |
| R2 Bits 15:8 DB1[7:0] | Data byte 1 |
| R2 Bits 7:0 D[7:0] | Data byte 0 |
| R3 Bits 31:24 DB7[7:0] | Data byte 7 |
| R3 Bits 23:16 DB6[7:0] | Data byte 6 |
60.3.6 FDCAN Tx buffer element

The Tx buffers section (three elements) can be configured to hold Tx FIFO or Tx queue. The
Tx handler distinguishes between Tx FIFO and Tx queue using the Tx buffer configuration
TFQM bit of the FDCAN_TXBC register. The element size is configured for storage of CAN
FD messages with up to 64-byte data.

Table 640. Rx FIFO element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>R3 Bits 15:8 DB5[7:0]</td>
<td>Data byte 5</td>
</tr>
<tr>
<td>R3 Bits 7:0 DB4[7:0]</td>
<td>Data byte 4</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>Rn Bits 31:24 DBm[7:0]</td>
<td>Data byte m</td>
</tr>
<tr>
<td>Rn Bits 23:16 DBm-1[7:0]</td>
<td>Data byte m-1</td>
</tr>
<tr>
<td>Rn Bits 15:8 DBm-2[7:0]</td>
<td>Data byte m-2</td>
</tr>
<tr>
<td>Rn Bits 7:0 DBm-3[7:0]</td>
<td>Data byte m-3</td>
</tr>
</tbody>
</table>

Table 641. Tx buffer and FIFO element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
<td>ID[28:0]</td>
<td>MM[7:0]</td>
<td>EFC</td>
<td>Res.</td>
</tr>
<tr>
<td>T2</td>
<td>DB3[7:0]</td>
<td>DB2[7:0]</td>
<td>DB1[7:0]</td>
<td>D[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>DB7[7:0]</td>
<td>DB6[7:0]</td>
<td>DB5[7:0]</td>
<td>DB4[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tn</td>
<td>DBm[7:0]</td>
<td>DBm-1[7:0]</td>
<td>DBm-2[7:0]</td>
<td>DBm-3[7:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 642. Tx buffer element description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0 Bit 31 ESI(1)</td>
<td>Error state indicator</td>
</tr>
<tr>
<td>– 0: ESI bit in CAN FD format depends only on error passive flag</td>
<td></td>
</tr>
<tr>
<td>– 1: ESI bit in CAN FD format transmitted recessive</td>
<td></td>
</tr>
<tr>
<td>T0 Bit 30 XTD</td>
<td>Extended identifier</td>
</tr>
<tr>
<td>– 0: 11-bit standard identifier</td>
<td></td>
</tr>
<tr>
<td>– 1: 29-bit extended identifier</td>
<td></td>
</tr>
</tbody>
</table>
Table 642. Tx buffer element description (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| T0 Bit 29 RTR\(^{(2)}\) | Remote transmission request  
  – 0: Transmit data frame  
  – 1: Transmit remote frame |
| T0 Bits 28:0 ID[28:0] | Identifier  
  Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |
| T1 Bits 31:24 MM[7:0] | Message marker  
  Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status. |
| T1 Bit 23 EFC | Event FIFO control  
  – 0: Do not store Tx events  
  – 1: Store Tx events |
| T1 Bit 21 FDF | FD format  
  – 0: Frame transmitted in classic CAN format  
  – 1: Frame transmitted in CAN FD format |
| T1 Bit 20 BRS\(^{(3)}\) | Bit rate switching  
  – 0: CAN FD frames transmitted without bit rate switching  
  – 1: CAN FD frames transmitted with bit rate switching |
| T1 Bits 19:16 DLC[3:0] | Data length code  
  – 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes  
  – 9 - 15: Classic CAN: received frame has 8 data bytes  
  – 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes |
| T2 Bits 31:24 DB3[7:0] | Data byte 3 |
| T2 Bits 23:16 DB2[7:0] | Data byte 2 |
| T2 Bits 15:8 DB1[7:0] | Data byte 1 |
| T2 Bits 7:0 D[7:0] | Data byte 0 |
| T3 Bits 31:24 DB7[7:0] | Data byte 7 |
| T3 Bits 23:16 DB6[7:0] | Data byte 6 |
| T3 Bits 15:8 DB5[7:0] | Data byte 5 |
| T3 Bits 7:0 DB4[7:0] | Data byte 4 |
| : | : |
60.3.7 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event, FIFO, the host CPU gets this information in the order that the messages were transmitted. Status information about the Tx event FIFO can be obtained from FDCAN_TXEFS register.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>XTD</th>
<th>RTR</th>
<th>ID[28:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0</td>
<td>ESI</td>
<td>XTD</td>
<td>RTR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| E0 Bit 31 | ESI Error state indicator  
  – 0: Transmitting node is error active  
  – 1: Transmitting node is error passive |
| E0 Bit 30 | XTD Extended identifier  
  – 0: 11-bit standard identifier  
  – 1: 29-bit extended identifier |
| E0 Bit 29 | RTR Remote transmission request  
  – 0: Transmit data frame  
  – 1: Transmit remote frame |
| E0 Bits 28:0 | ID[28:0] Identifier  
  Standard or extended identifier depending on bit XTD. A standard identifier has to be written to ID[28:18]. |
60.3.8 FDCAN standard message ID filter element

Up to 28 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FLSSA plus the index of the filter element (0 … 27).

**Table 645. Standard message ID filter element**

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
</table>
Note: In case a reserved value is configured, the filter element is considered disabled.

### 60.3.9 FDCAN extended message ID filter element

Up to eight filter elements can be configured for 29-bit extended IDs. When accessing an extended message ID filter element, its address is the filter list extended start address FLESA plus twice the index of the filter element (0 ... 7).

#### Table 647. Extended message ID filter element

<table>
<thead>
<tr>
<th>Bit</th>
<th>31</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0</td>
<td></td>
<td>EFEC[2:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1</td>
<td></td>
<td>ELTI[1:0]</td>
<td>Res.</td>
<td></td>
<td>EFID2[28:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| F0 Bits 31:29 EFEC[2:0] | Extended filter element configuration  
Extended filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC[2:0] = 100, 101 or 110 a match sets interrupt flag IR[HPM] and, if enabled, an interrupt is generated. In this case register HPMS is updated with the status of the priority match.  
– 000: Disable filter element  
– 001: Store in Rx FIFO 0 if filter matches  
– 010: Store in Rx FIFO 1 if filter matches  
– 011: Reject ID if filter matches  
– 100: Set priority if filter matches  
– 101: Set priority and store in FIFO 0 if filter matches  
– 110: Set priority and store in FIFO 1 if filter matches  
– 111: Not used |
| F0 Bits 28:0 EFID1[28:0] | Extended filter ID 1  
First ID of extended ID filter element. When filtering for Rx FIFO, this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only XIDAM masking mechanism. |
| F1 Bits 31:30 EFT[1:0] | Extended filter type  
– 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID)  
– 01: Dual ID filter for EF1ID or EF2ID  
– 10: Classic filter: EF1ID = filter, EF2ID = mask  
– 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), XIDAM mask not applied |
| F1 Bit 29 | Not used |
| F1 Bits 28:0 EFID2[28:0] | Extended filter ID 2  
Second ID of extended ID filter element. |
60.4 FDCAN registers

60.4.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000
Reset value: 0x3214 1218

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r</td>
<td>r r r r</td>
<td>r r r r</td>
<td>r r r r</td>
</tr>
<tr>
<td>15 14 13 12</td>
<td>11 10 9 8</td>
<td>7 6 5 4</td>
<td>3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:28 REL[3:0]: 3
Bits 27:24 STEP[3:0]: 2
Bits 23:20 SUBSTEP[3:0]: 1
Bits 19:16 YEAR[3:0]: 4
Bits 15:8 MON[7:0]: 12
Bits 7:0 DAY[7:0]: 18

60.4.2 FDCAN endian register (FDCAN_ENDN)

Address offset: 0x0004
Reset value: 0x8765 4321

<table>
<thead>
<tr>
<th>ETV[31:16]</th>
<th>ETV[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r</td>
<td>r r r r</td>
</tr>
<tr>
<td>15 14 13 12</td>
<td>11 10 9 8</td>
</tr>
</tbody>
</table>

Bits 31:0 ETV[31:0]: Endianness test value
The endianness test value is 0x8765 4321.

Note: The register read must give the reset value to ensure no endianness issue.

60.4.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C
Reset value: 0x0000 0A33

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR are set. The CAN time quantum can be programmed in the range of 1 to 32 FDCAN clock periods:
\[ t_q = (DBRP[4:0] + 1) \text{ FDCAN clock periods} \]
DTSEG1[4:0] is the sum of PROPSEG and PHASE_SEG1. DTSEG2[3:0] is PHASE_SEG2. Therefore, the length of the bit time is (programmed values) × [DTSEG1[4:0] + DTSEG2[3:0] + 3] × t_q or (functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

Note: With an FDCAN clock of 8 MHz, the reset value 0x0000 0A33 configures the FDCAN for a fast bit rate of 500 kbit/s. The data phase bit rate must be higher than or equal to the nominal bit rate.

### 60.4.4 FDCAN test register (FDCAN_TEST)

Write access to this register is enabled by setting the TEST bit of the FDCAN_CCCR register. All register functions are set to their reset values when this bit is cleared.

Loop-back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX[1:0] differently from 00 can disturb the message transfer on the CAN bus.

Address offset: 0x0010
Reset value: 0x0000 0000
60.4.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured through the WDC[7:0] bitfield of the FDCAN_RWD register.

The counter is reloaded with WDC[7:0] when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops, and the interrupt flag WDI is set in the FDCAN_IR register. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014
Reset value: 0x0000 0000
60.4.6 **FDCAN CC control register (FDCAN_CCCR)**

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and clearing single bits, see *Software initialization*.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>NISO</td>
<td>TXP</td>
<td>EFBi</td>
<td>PXHD</td>
<td>BRSE</td>
<td>FDOE</td>
<td>TEST</td>
<td>DAR</td>
<td>MON</td>
<td>CSR</td>
<td>CSA</td>
<td>ASM</td>
<td>CCE</td>
<td>INIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

**Bit 15 NISO**: Non-ISO operation
- If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
  - 0: CAN FD frame format according to ISO11898-1
  - 1: CAN FD frame format according to Bosch CAN FD Specification V1.0

**Bit 14 TXP**: Transmit pause enable
- If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
  - 0: Disabled
  - 1: Enabled

**Bit 13 EFBi**: Edge filtering during bus integration
- 0: Edge filtering disabled
- 1: Two consecutive dominant t_q required to detect an edge for hard synchronization

**Bit 12 PXHD**: Protocol exception handling disable
- 0: Protocol exception handling enabled
- 1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

**Bit 9 BRSE**: FDCAN bit rate switching
- 0: Bit rate switching for transmissions disabled
- 1: Bit rate switching for transmissions enabled

**Bit 8 FDOE**: FD operation enable
- 0: FD operation disabled
- 1: FD operation enabled

**Bit 7 TEST**: Test mode enable
- 0: Normal operation, FDCAN_TEST holds reset values
- 1: Test mode, write access to FDCAN_TEST enabled
Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
This bit can only be set by software when both CCE and INIT are set. The bit can be cleared by the host at any time.
0: Bus monitoring mode disabled
1: Bus monitoring mode enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus is idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN can be set in power-down by stopping APB clock and kernel clock.

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bit rates. The application tests different bit rates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it wait for the occurrence of bus-idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can only be set by software when both CCE and INIT are set. The bit can be cleared by the software at any time.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers.
1: The CPU has write access to the protected configuration registers (while INIT set in FDCAN_CCCR).

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization started

**Note:** Due to the synchronization mechanism between the two clock domains, there can be a delay until the value written to INIT can be read back. Therefore, the programmer has to assure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

### 60.4.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C
Reset value: 0x0600 0A03

This register is only writable if the CCE and INIT bits of the FDCAN_CCCR register are both set. The CAN bit time can be programmed in the range of 4 to 81 × tq. The CAN time quantum can be programmed in the range of 1 to 1024 FDCAN kernel clock periods:

\[ t_q = (BRP + 1) \times \text{FDCAN clock period fdcan_ker_ck} \]
NTSEG1[7:0] is the sum of PROP_SEG and PHASE_SEG1. NTSEG2[6:0] is PHASE_SEG2. Therefore, the length of the bit time is (programmed values) × [NTSEG1[7:0] + NTSEG2[6:0] + 3] × t_q or (functional values) × [SYNC_SEG + PROP_SEG + PHASE_SEG1 + PHASE_SEG2] × t_q.

The information processing time (IPT) is 0, meaning the data for the next bit is available at the first clock edge after the sample point.

Note: With a CAN kernel clock of 48 MHz, the reset value of 0x0600 0A03 configures the FDCAN for a bit rate of 3 Mbit/s.
60.4.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19:16</td>
<td><strong>TCP[3:0]</strong>: Timestamp counter prescaler</td>
</tr>
<tr>
<td></td>
<td>Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1…16].</td>
</tr>
<tr>
<td></td>
<td>The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.</td>
</tr>
<tr>
<td></td>
<td>In CAN FD mode, the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS[1:0] = 10).</td>
</tr>
<tr>
<td></td>
<td>This bitfield is write-protected (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.</td>
</tr>
<tr>
<td>15:2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1:0</td>
<td><strong>TSS[1:0]</strong>: Timestamp select</td>
</tr>
<tr>
<td>00</td>
<td>Timestamp counter value always 0x0000</td>
</tr>
<tr>
<td>01</td>
<td>Timestamp counter value incremented according to TCP</td>
</tr>
<tr>
<td>10</td>
<td>External timestamp counter from TIM3 value (tim3_cnt[0:15])</td>
</tr>
<tr>
<td>11</td>
<td>Same as 00.</td>
</tr>
<tr>
<td></td>
<td>These bits are write-protected write (P): write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.</td>
</tr>
</tbody>
</table>

60.4.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
60.4.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028
Reset value: 0xFFFF 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 **TOP[15:0]**: Timeout period
Start value of the timeout counter (down-counter). Configures the timeout period.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select
When operating in continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by TOP[15:0] in FDCAN_TOCC and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by TOP[15:0]. Down-counting is started when the first FIFO element is stored.

00: Continuous operation
01: Timeout controlled by Tx event FIFO
10: Timeout controlled by Rx FIFO 0
11: Timeout controlled by Rx FIFO 1

This bitfield is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 0 **ETOC**: Timeout counter enable
0: Timeout counter disabled
1: Timeout counter enabled

This bit is write-protected (P), write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

For more details, see **Timeout counter**.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 that is not caused by write access to FDCAN_TSCV.
60.4.11  FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C
Reset value: 0x0000 FFFF

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:0  | TOC[15:0]: Timeout counter |
|           | The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of the TCP[3:0] bitfield of the FDCAN_TSCC register. When decremented to 0, the TOO interrupt flag is set in FDCAN_IR and the timeout counter is stopped. Start and reset/restart conditions are configured via TOS[1:0] in FDCAN_TOCC. |

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

rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w rc_w

60.4.12  FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040
Reset value: 0x0000 0000

| Bits 31:24 | Reserved, must be kept at reset value. |
| Bits 23:16 | CEL[7:0]: CAN error logging |
|           | The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL[7:0]. The counter stops at 0xFF; the next increment of TEC[7:0] or REC[6:0] sets the ELO interrupt flag in FDCAN_IR. Access type is rc_r: cleared on read. |
| Bit 15     | RP: Receive error passive |
|           | 0: The receive error counter is below the error passive level of 128. |
|           | 1: The receive error counter has reached the error passive level of 128. |
| Bits 14:8  | REC[6:0]: Receive error counter |
|           | Actual state of the receive error counter, values between 0 and 127. |
| Bits 7:0   | TEC[7:0]: Transmit error counter |
|           | Actual state of the transmit error counter, values between 0 and 255. |
|           | When the ASM bit of the FDCAN_CCCR is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL[7:0] is still incremented. |

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

rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r rc_r

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

r r r r r r r r r r r r r r r r

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
r r r r r r r r r r r r r r r r
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
TEC[7:0]
60.4.13  FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044
Reset value: 0x0000 0707

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 22:16  **TDCV[6:0]:** Transmitter delay compensation value
Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and TDCO[6:0] in FDCAN_TDCR. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 × mtq.

Bit 15  Reserved, must be kept at reset value.

Bit 14  **PXE:** Protocol exception event
0: No protocol exception event occurred since last read access
1: Protocol exception event occurred

Bit 13  **REDL:** Received FDCAN message
This bit is set independent of acceptance filtering.
0: Since this bit was cleared by the CPU, no FDCAN message has been received.
1: Message in FDCAN format with EDL flag set has been received.
Access type is rc_r: cleared on read.

Bit 12  **RBRS:** BRS flag of last received FDCAN message
This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not have its BRS flag set.
1: Last received FDCAN message had its BRS flag set.
Access type is rc_r: cleared on read.

Bit 11  **RESI:** ESI flag of last received FDCAN message
This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not have its ESI flag set.
1: Last received FDCAN message had its ESI flag set.
Access type is rc_r: cleared on read.

Bits 10:8  **DLEC[2:0]:** Data last error code
Type of last error that occurred in the data phase of a FDCAN format frame with its BRS flag set. Coding is the same as for LEC[2:0]. This field is cleared when a FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.
Access type is rs: set on read.

Bit 7  **BO:** Bus-off status
0: The FDCAN is not in bus-off state.
1: The FDCAN is in bus-off state.
Bit 6  **EW**: Warning status  
0: Both error counters are below the error-warning limit of 96.  
1: At least one of error counter has reached the error-warning limit of 96.

Bit 5  **EP**: Error passive  
0: The FDCAN is in the error-active state. It normally takes part in bus communication and sends an active error flag when an error has been detected.  
1: The FDCAN is in the error-passive state.

**Bits 4:3 ACT[1:0]**: Activity
Monitors the module’s CAN communication state.
00: Synchronizing: node is synchronizing on CAN communication.  
01: Idle: node is neither receiver nor transmitter.  
10: Receiver: node is operating as receiver.  
11: Transmitter: node is operating as transmitter.

**Bits 2:0 LEC[2:0]**: Last error code  
LEC[2:0] indicates the type of the last error to occur on the CAN bus. This bitfield is cleared when a message has been transferred (reception or transmission) without error.  
000: No error occurred since LEC[2:0] has been cleared by successful reception or transmission.  
001: Stuff error. More than five equal bits in a sequence have occurred in a part of a received message where this is not allowed.  
010: Form error. A fixed format part of a received frame has the wrong format.  
011: Ack error. The message transmitted by the FDCAN was not acknowledged by another node.  
100: Bit1 error. During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.  
101: Bit0 error. During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During bus-off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).  
110: CRC error. The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.  
111: No change. Any read access to the protocol status register reinitializes LEC[2:0] to 7. When the LEC[2:0] shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.

**Note:** When a frame in FDCAN format has reached the data phase with the BRS flag set, the next CAN event (error or valid frame) is shown in DLEC[2:0] instead of LEC[2:0]. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a form error, not as a stuff error.

The bus-off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or clearing the INIT bit of the FDCAN_CCCR register. If the device enters bus-off, it sets the INIT bit of its own, stopping all bus activities. Once INIT has been cleared by the CPU, the device waits for 129 occurrences of bus-idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the bus-off recovery sequence, the error management counters are reset. During the waiting time after clearing INIT, each time a sequence of 11 recessive bits has been monitored, a bit0 error code is written to LEC[2:0] of FDCAN_PSR, enabling the CPU to check up whether the CAN bus is...
stuck at dominant or continuously disturbed, and to monitor the bus-off recovery sequence. The \text{REC}[6:0] bitfield of the FDCAN\_ECR register is used to count these sequences.

60.4.14 **FDCAN transmitter delay compensation register (FDCAN\_TDCR)**

Address offset: 0x0048

Reset value: 0x0000 0000

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN\_TX to FDCAN\_RX and the secondary sample point. Valid values are 0 to 127 × mtq.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN\_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN\_CCCR register are both set.

60.4.15 **FDCAN interrupt register (FDCAN\_IR)**

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the host clears them. A flag is cleared by writing 1 to the corresponding bit position.

Writing 0 has no effect. A hard reset clears the register. The configuration of FDCAN\_IE controls whether an interrupt is generated. The configuration of FDCAN\_ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

Bits 31:24 Reserved, must be kept at reset value.
Bit 23 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 22 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (DLEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 21 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (LEC[2:0] different from 0 and 7 in FDCAN_PSR)

Bit 20 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 19 **BO**: Bus-off status
0: Bus-off status unchanged
1: Bus-off status changed

Bit 18 **EW**: Warning status
0: Error-warning status unchanged
1: Error-warning status changed

Bit 17 **EP**: Error passive
0: Error-passive status unchanged
1: Error-passive status changed

Bit 16 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow.
1: Overflow of CAN error logging counter occurred.

Bit 15 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 14 **MRAF**: Message RAM access failure
The flag is set when the Rx handler:
- has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated. The partly stored message is overwritten when the next message is stored to this location.

The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure, the FDCAN is switched into restricted operation mode (see Restricted operation mode). To leave restricted operation mode, the host CPU has to clear the ASM of the FDCAN_CCCR register.
0: No message RAM access failure occurred
1: Message RAM access failure occurred
Bit 13 **TSW**: Timestamp wraparound
0: No timestamp counter wrap-around
1: Timestamp counter wrapped around

Bit 12 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost

Bit 11 **TEFF**: Tx event FIFO full
0: Tx event FIFO Not full
1: Tx event FIFO full

Bit 10 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element.

Bit 9 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 8 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 7 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 6 **HPM**: High-priority message
0: No high-priority message received
1: High-priority message received

Bit 5 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost

Bit 4 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 3 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 2 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost

Bit 1 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 0 **RF0N**: Rx FIFO 0 new message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0
### 60.4.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register are signaled on an interrupt line.

**Address offset:** 0x0054

**Reset value:** 0x0000 0000

| Bit 31:24 Reserved, must be kept at reset value. |
| Bit 23 **ARAE**: Access to reserved address enable |
| Bit 22 **PEDE**: Protocol error in data phase enable |
| Bit 21 **PEAE**: Protocol error in arbitration phase enable |
| Bit 20 **WDIE**: Watchdog interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 19 **BOE**: Bus-off status |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 18 **EWE**: Warning status interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 17 **EPE**: Error passive interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 16 **ELOE**: Error logging overflow interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 15 **TOOE**: Timeout occurred interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 14 **MRAFE**: Message RAM access failure interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 13 **TSWE**: Timestamp wraparound interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
| Bit 12 **TEFLE**: Tx event FIFO element lost interrupt enable |
| | 0: Interrupt disabled |
| | 1: Interrupt enabled |
Bit 11  **TEFFE**: Tx event FIFO full interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 10  **TEFNE**: Tx event FIFO new entry interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 9   **TFEE**: Tx FIFO empty interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 8   **TCFE**: Transmission cancellation finished interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 7   **TCE**: Transmission completed interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 6   **HPME**: High-priority message interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 5   **RF1LE**: Rx FIFO 1 message lost interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 4   **RF1FE**: Rx FIFO 1 full interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 3   **RF1NE**: Rx FIFO 1 new message interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 2   **RF0LE**: Rx FIFO 0 message lost interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 1   **RF0FE**: Rx FIFO 0 full interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled

Bit 0   **RF0NE**: Rx FIFO 0 new message interrupt enable
       0: Interrupt disabled
       1: Interrupt enabled
### 60.4.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific group of interrupt flags from the interrupt register to one of the two module interrupt lines. For interrupt generation, the respective interrupt line has to be enabled via the EINT0 and EINT1 bit of the FDCAN_ILE register.

Address offset: 0x0058

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:7 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 6 <strong>PERR</strong>: Protocol error grouping the following interruption</td>
</tr>
<tr>
<td>ARAL: Access to reserved address line</td>
</tr>
<tr>
<td>PEDL: Protocol error in data phase line</td>
</tr>
<tr>
<td>PEAL: Protocol error in arbitration phase line</td>
</tr>
<tr>
<td>WDIL: Watchdog interrupt line</td>
</tr>
<tr>
<td>BOL: Bus-off status</td>
</tr>
<tr>
<td>EWL: Warning status interrupt line</td>
</tr>
<tr>
<td>Bit 5 <strong>BERR</strong>: Bit and line error grouping the following interruption</td>
</tr>
<tr>
<td>EPL: Error passive interrupt line</td>
</tr>
<tr>
<td>ELOL: Error logging overflow interrupt line</td>
</tr>
<tr>
<td>Bit 4 <strong>MISC</strong>: Interrupt regrouping the following interruption</td>
</tr>
<tr>
<td>TOOL: Timeout occurred interrupt line</td>
</tr>
<tr>
<td>MRAFL: Message RAM access failure interrupt line</td>
</tr>
<tr>
<td>TSWL: Timestamp wraparound interrupt line</td>
</tr>
<tr>
<td>Bit 3 <strong>TFERR</strong>: Tx FIFO ERROR grouping the following interruption</td>
</tr>
<tr>
<td>TEFLL: Tx event FIFO element lost interrupt line</td>
</tr>
<tr>
<td>TEFFL: Tx event FIFO full interrupt line</td>
</tr>
<tr>
<td>TEFNL: Tx event FIFO new entry interrupt line</td>
</tr>
<tr>
<td>TFEL: Tx FIFO empty interrupt line</td>
</tr>
<tr>
<td>Bit 2 <strong>SMSG</strong>: Status message bit grouping the following interruption</td>
</tr>
<tr>
<td>TCFL: Transmission cancellation finished interrupt line</td>
</tr>
<tr>
<td>TCL: Transmission completed interrupt line</td>
</tr>
<tr>
<td>HPML: High-priority message interrupt line</td>
</tr>
<tr>
<td>Bit 1 <strong>RXFIFO1</strong>: RX FIFO bit grouping the following interruption</td>
</tr>
<tr>
<td>RF1LL: Rx FIFO 1 message lost interrupt line</td>
</tr>
<tr>
<td>RF1FL: Rx FIFO 1 full interrupt line</td>
</tr>
<tr>
<td>RF1NL: Rx FIFO 1 new message interrupt line</td>
</tr>
</tbody>
</table>
### 60.4.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming the EINT0 and EINT1 bits.

Address offset: 0x005C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXFIFO0: RX FIFO bit grouping the following interruption</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>RF0LL: Rx FIFO 0 message lost interrupt line</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>RF0FL: Rx FIFO 0 full interrupt line</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>RF0NL: Rx FIFO 0 new message interrupt line</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>EINT1</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>EINT0</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 1 **EINT1**: Enable interrupt line 1
- 0: Interrupt line fdcan_intr0_it disabled
- 1: Interrupt line fdcan_intr0_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
- 0: Interrupt line fdcan_intr1_it disabled
- 1: Interrupt line fdcan_intr1_it enabled

### 60.4.19 FDCAN global filter configuration register (FDCAN_RXGFC)

Global settings for message ID filtering. The global filter configuration controls the filter path for standard and extended messages as described in Figure 890 and Figure 891.

Address offset: 0x0080

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RXFIFO0: RX FIFO bit grouping the following interruption</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>LSE[3:0]</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>LSS[4:0]</td>
<td>rw</td>
</tr>
<tr>
<td>28</td>
<td>FOM</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>F1OM</td>
<td>rw</td>
</tr>
<tr>
<td>26</td>
<td>ANFS[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>ANFE[1:0]</td>
<td>rw</td>
</tr>
<tr>
<td>24</td>
<td>RRFS</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>RRFE</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.
Bits 27:24 **LSE[3:0]**: Number of extended filter elements in the list
- 0: No extended message ID filter
- 1 to 8: Number of extended message ID filter elements
- > 8: Values greater than 8 are interpreted as 8.
  
  This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **LSS[4:0]**: Number of standard filter elements in the list
- 0: No standard message ID filter
- 1 to 28: Number of standard message ID filter elements
- > 28: Values greater than 28 are interpreted as 28.
  
  This bitfield is write protected (P), which means that write access by the bits is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **FOM**: FIFO 0 operation mode (overwrite or blocking)
  
  This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 8 **F1OM**: FIFO 1 operation mode (overwrite or blocking)
  
  This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept Non-matching frames standard
  
  Defines how received messages with 11-bit IDs that do not match any element of the filter list are treated.
  - 00: Accept in Rx FIFO 0
  - 01: Accept in Rx FIFO 1
  - 10: Reject
  - 11: Reject
  
  This bitfield is write-protected (P), which means write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
  
  Defines how received messages with 29-bit IDs that do not match any element of the filter list are treated.
  - 00: Accept in Rx FIFO 0
  - 01: Accept in Rx FIFO 1
  - 10: Reject
  - 11: Reject
  
  This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bit 1 **RRFS**: Reject remote frames standard
  
  - 0: Filter remote frames with 11-bit standard IDs
  - 1: Reject all remote frames with 11-bit standard IDs
  
  This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.
60.4.20  FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0084
Reset value: 0x1FFF FFFF

| Bit 31:29 | Reserved, must be kept at reset value. |
| Bit 28:0  | EIDM[28:0]: Extended ID mask |
|           | For acceptance filtering of extended frames the extended ID AND mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set, the mask is not active. |
|           | This bitfield is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set. |

60.4.21  FDCAN high-priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0088
Reset value: 0x0000 0000

| Bit 31:16 | Reserved, must be kept at reset value. |
| Bit 15    | FLST: Filter list |
|           | Indicates the filter list of the matching filter element: |
|           | 0: Standard filter list |
|           | 1: Extended filter list |
| Bit 14:13 | Reserved, must be kept at reset value. |
Bits 12:8 **FIDX[4:0]:** Filter index  
Index of matching filter element.  
Range: 0 to LSS[4:0] - 1 or LSE[3:0] - 1 in FDCAN_RXGFC.

Bits 7:6 **MSI[1:0]:** Message storage indicator  
00: No FIFO selected  
01: FIFO overrun  
10: Message stored in FIFO 0  
11: Message stored in FIFO 1

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **BIDX[2:0]:** Buffer index  
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

**60.4.22 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)**

Address offset: 0x0090  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RF0L</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>F0F</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>F0PI[1:0]</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>F0FL[3:0]</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 25 **RF0L:** Rx FIFO 0 message lost  
This bit is a copy of the RF0L interrupt flag of the FDCAN_IR register. When RF0L is cleared, this bit is also cleared.  
0: No Rx FIFO 0 message lost  
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F:** Rx FIFO 0 full  
0: Rx FIFO 0 not full  
1: Rx FIFO 0 full

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **F0PI[1:0]:** Rx FIFO 0 put index  
Rx FIFO 0 write index pointer.  
Range: 0 to 2.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **F0GI[1:0]:** Rx FIFO 0 get index  
Rx FIFO 0 read index pointer.  
Range: 0 to 2.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **F0FL[3:0]:** Rx FIFO 0 fill level  
Number of elements stored in Rx FIFO 0.  
Range: 0 to 3.
### 60.4.23 CAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x0094  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:3 Reserved, must be kept at reset value.  
Bits 2:0 **F0AI[2:0]**: Rx FIFO 0 acknowledge index  
After the host has read a message or a sequence of messages from Rx FIFO 0, it has to write the buffer index of the last element read from Rx FIFO 0 to F0AI[2:0]. This sets the Rx FIFO 0 get index (F0GI[1:0] of FDCAN_RXF0S) to F0AI[2:0] + 1 and updates the FIFO 0 fill level (F0FL[3:0] FDCAN_RXF0S).

### 60.4.24 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x0098  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:26 Reserved, must be kept at reset value.  
Bit 25 **RF1L**: Rx FIFO 1 message lost  
This bit is a copy of the RF1L interrupt flag of the FDCAN_IR register. When RF1L is cleared, this bit is also cleared.  
0: No Rx FIFO 1 message lost  
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0  
Bit 24 **F1F**: Rx FIFO 1 full  
0: Rx FIFO 1 not full  
1: Rx FIFO 1 full  
Bits 23:18 Reserved, must be kept at reset value.  
Bits 17:16 **F1PI[1:0]**: Rx FIFO 1 put index  
Rx FIFO 1 write index pointer.  
Range: 0 to 2.  
Bits 15:10 Reserved, must be kept at reset value.
Bits 9:8  **F1GI[1:0]**: Rx FIFO 1 get index
RxFIFO 1 read index pointer.
Range: 0 to 2.

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  **F1FL[3:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1.
Range: 0 to 3.

### 60.4.25 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x009C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 2:0  **F1AI[2:0]**: Rx FIFO 1 acknowledge index
After the host has read a message or a sequence of messages from Rx FIFO 1, it has to write the buffer index of the last element read from Rx FIFO 1 to F1AI[2:0]. This sets the Rx FIFO 1 get index (F1GI[1:0] of FDCAN_RXF1S) to F1AI[2:0] + 1 and updates the FIFO 1 fill level (F1FL[3:0] FDCAN_RXF1S).

### 60.4.26 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:25  Reserved, must be kept at reset value.

Bit 24  **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
This bit is write-protected (P), which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

Bits 23:0  Reserved, must be kept at reset value.
60.4.27  FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in the FDCAN_TXBRP register. Therefore, the effect of add/cancellation requests can be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 21  **TFQF**: Tx FIFO/queue full
0: Tx FIFO/queue not full
1: Tx FIFO/queue full

Bits 20:18  Reserved, must be kept at reset value.

Bits 17:16  **TFQPI[1:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 3

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:8  **TFGI[1:0]**: Tx FIFO get index
Tx FIFO read index pointer, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC)

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

Bits 2:0  **TFFL[2:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 3. Read as 0 when Tx queue operation is configured (TFQM = 1 in FDCAN_TXBC).

60.4.28  FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00C8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:3  Reserved, must be kept at reset value.
Bits 2:0 **TRP[2:0]**: Transmission request pending
Each Tx buffer has its own transmission request pending bit. The bits are set via the FDCAN_TXBAR register. The bits are cleared after a requested transmission has completed or has been canceled via the FDCAN_TXBCR register.

After the FDCAN_TXBRP bit has been set, a Tx scan is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of the FDCAN_TXBRP register. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are directly cleared after the corresponding FDCAN_TXBRP bit has been cleared.

After a cancellation has been requested, a finished cancellation is signaled via the FDCAN_TXBCF in the following cases:
- after successful transmission together with the corresponding TXBTO bit
- when the transmission has not yet been started at the point of cancellation
- when the transmission has been aborted due to lost arbitration
- when an error occurred during frame transmission

In DAR mode, all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending
1: Transmission request pending

**Note:** FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately. The corresponding FDCAN_TXBRP bit is cleared.

### 60.4.29 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00CC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 2:0 **AR[2:0]**: Add request
Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the host to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. When no Tx scan is running, the bits are cleared immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added
1: Transmission requested added.

**Note:** If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.
### 60.4.30 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 2:0  **CR[2:0]: Cancellation request**

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding CR bit; writing a 0 has no impact. This enables the host to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. The bits remain set until the corresponding FDCAN_TXBRP bit is cleared.

0: No cancellation pending  
1: Cancellation pending

### 60.4.31 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 2:0  **TO[2:0]: Transmission occurred.**

Each Tx buffer has its own TO bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred  
1: Transmission occurred
60.4.32  FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00D8
Reset value: 0x0000 0000

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

Bits 2:0  CF[2:0]: Cancellation finished

Each Tx buffer has its own CF bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are cleared when a new transmission is requested by writing a 1 to the corresponding bit of the FDCAN_TXBAR register.

0: No transmit buffer cancellation
1: Transmit buffer cancellation finished

60.4.33  FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00DC
Reset value: 0x0000 0000

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

Bits 2:0  TIE[2:0]: Transmission interrupt enable

Each Tx buffer has its own TIE bit.

0: Transmission interrupt disabled
1: Transmission interrupt enable
### 60.4.34 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| Bits 31:3 | Reserved, must be kept at reset value.  
| Bits 2:0 | **CFIE[2:0]**: Cancellation finished interrupt enable.  
| | Each Tx buffer has its own CFIE bit.  
| | 0: Cancellation finished interrupt disabled  
| | 1: Cancellation finished interrupt enabled

### 60.4.35 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00E4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| Bits 31:26 | Reserved, must be kept at reset value.  
| Bit 25 | **TEFL**: Tx event FIFO element lost  
| | This bit is a copy of the TEFL interrupt flag of the FDCAN_IR. When TEFL is cleared, this bit is also cleared.  
| | 0: No Tx event FIFO element lost  
| | 1: Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.  
| Bit 24 | **EFF**: Event FIFO full  
| | 0: Tx event FIFO not full  
| | 1: Tx event FIFO full  
| Bits 23:18 | Reserved, must be kept at reset value.  
| Bits 17:16 | **EFPI[1:0]**: Event FIFO put index  
| | Tx event FIFO write index pointer.  
| | Range: 0 to 3.  
| Bits 15:10 | Reserved, must be kept at reset value.
### 60.4.36 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

**Address offset:** 0x00E8  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bits 31:2**: Reserved, must be kept at reset value.
- **Bits 1:0** EFAI[1:0]: Event FIFO acknowledge index  
  After the host has read an element or a sequence of elements from the Tx event FIFO, it has to write the index of the last element read from Tx event FIFO to EFAI[1:0]. This sets the Tx event FIFO get index (EFGI[1:0] of FDCAN_TXEFS) to EFAI[1:0] + 1 and updates the FIFO fill level (EFFL[2:0] of FDCAN_TXEFS).

### 60.4.37 FDCAN CFG clock divider register (FDCAN_CKDIV)

**Address offset:** 0x0100  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Bits 31:4**: Reserved, must be kept at reset value.
Bits 3:0 **PDIV[3:0]**: input clock divider

The CAN kernel clock can be divided prior to be used by the CAN subsystem. The rate must be computed using the divider output clock.

- 0000: Divide by 1
- 0001: Divide by 2
- 0010: Divide by 4
- 0011: Divide by 6
- 0100: Divide by 8
- 0101: Divide by 10
- 0110: Divide by 12
- 0111: Divide by 14
- 1000: Divide by 16
- 1001: Divide by 18
- 1010: Divide by 20
- 1011: Divide by 22
- 1100: Divide by 24
- 1101: Divide by 26
- 1110: Divide by 28
- 1111: Divide by 30

This bitfield is write-protected (P): which means that write access is possible only when the CCE and INIT bits of the FDCAN_CCCR register are both set.

*Note: The clock divider is common to all FDCAN instances. Only FDCAN1 instance has FDCAN_CKDIV register, which changes clock divider for all instances.*

### 60.4.38 FDCAN register map

Table 649. FDCAN register map and reset values

<p>| Offset   | Register          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|          | Reset value       | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0004   | FDCAN_ENDN        | ETV[31:0] | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
|          | Reset value       | 1  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 1  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0008   | Reserved          | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| 0x000C   | FDCAN_DBTP        | TDC | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
|          | Reset value       | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x0010   | FDCAN_TEST        | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
|          | Reset value       | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
|          | Reset value       | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x001B</td>
<td>FDCAN_CCCR</td>
<td>0x001C</td>
<td>FDCAN_NBTP</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0020</td>
<td>FDCAN_TSCC</td>
<td>0x0022</td>
<td>FDCAN_TSCV</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>TSC[15:0]</td>
</tr>
<tr>
<td>0x0024</td>
<td>FDCAN_TSCV</td>
<td>0x0028</td>
<td>FDCAN_TOCC</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>TOC[15:0]</td>
</tr>
<tr>
<td>0x0030</td>
<td>Reserved</td>
<td>0x003C</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0040</td>
<td>FDCAN_ECR</td>
<td>0x0054</td>
<td>FDCAN_IE</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0044</td>
<td>FDCAN_PSR</td>
<td>0x0058</td>
<td>FDCAN_ILS</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0048</td>
<td>FDCAN_UDCR</td>
<td>0x005C</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Table 649. FDCAN register map and reset values (continued)
### Table 649. FDCAN register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
<th>Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x005C</td>
<td>FDCAN_ILE</td>
<td>0x0060</td>
<td>Reserved</td>
<td>0x007C</td>
<td>Reserved</td>
<td>0x0080</td>
<td>FDCAN_RXGFC</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0084</td>
<td>FDCAN_XIDAM</td>
<td>0x0088</td>
<td>FDCAN_HPMS</td>
<td>0x0090</td>
<td>FDCAN_RXF0S</td>
<td>0x0094</td>
<td>FDCAN_RXF0A</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>LSIS1</td>
<td></td>
<td>F1GI[1:0]</td>
<td></td>
<td>F1AI[2:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F0OM</td>
<td></td>
<td>F1FL[2:0]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>F0OM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>FDCAN_TXBC</td>
<td>0x00C4</td>
<td>FDCAN_TXF0S</td>
<td>0x00CB</td>
<td>FDCAN_TXF1S</td>
<td>0x00D0</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Reset Values

- **FDCAN_ILE**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_RXGFC**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_XIDAM**:
  - EIDM[28:0]: 0
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_HPMS**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_RXF0S**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_RXF0A**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_RXF1S**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0

- **FDCAN_RXF1A**:
  - LSE[3:0]: 0 0 0 0
  - LSIS1: 0
  - F0OM: 0
  - F1OM: 0
  - R0: 0
  - R1: 0
  - R2: 0
  - R3: 0
  - R4: 0
  - R5: 0
  - R6: 0
  - R7: 0
  - EN[1:0]: 0 0
  - ANFE[1:0]: 0 0
  - RRFS: 0
  - RRFE: 0
### Table 649. FDCAN register map and reset values (continued)

| Offset | Register          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0          |
|--------|-------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00CC | FDCAN _TXBAR      |    |    |    |    |    |    |    |    |    |    |    | AR3 | AR2 | AR1 | AR0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00D0 | FDCAN _TXBCR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00D4 | FDCAN _TXBTO      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00D8 | FDCAN _TXBCF      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00DC | FDCAN _TXBTIE     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00E0 | FDCAN _TXBCIE     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00E4 | FDCAN _TXEFS      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00E8 | FDCAN _TXEFA      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x0100 | FDCAN _CKDIV      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                   |    |    |    |    |    |    |    |    |    |    |    | 0   | 0   | 0   | 0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to **Section 2.3 on page 149** for the register boundary addresses.
USB on-the-go full-speed (OTG_FS)

Introduction

This section presents the architecture and the programming model of the OTG_FS controller.

The following acronyms are used throughout the section:
- FS: Full-speed
- LS: Low-speed
- MAC: Media access controller
- OTG: On-the-go
- PFC: Packet FIFO controller
- PHY: Physical layer
- USB: Universal serial bus
- UTMI: USB 2.0 Transceiver Macrocell interface (UTMI)
- LPM: Link power management
- BCD: Battery charging detector
- HNP: Host negotiation protocol
- SRP: Session request protocol

References are made to the following documents:
- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the On-The-Go Supplement to the USB 2.0 Specification. It can also be configured as a host-only or device-only controller, fully compliant with the USB 2.0 Specification. OTG_FS supports the speeds defined in the Table 650: OTG_FS speeds supported below. The only external device required is a charge pump for VBUS in OTG mode.

<table>
<thead>
<tr>
<th>Mode</th>
<th>HS (480 Mb/s)</th>
<th>FS (12 Mb/s)</th>
<th>LS (1.5 Mb/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host mode</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Device mode</td>
<td>-</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>
61.2 OTG_FS main features

The main features can be divided into three categories: general, host-mode and device-mode features.

61.2.1 General features

The OTG_FS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0
- OTG_FS supports the following PHY interface:
  - An on-chip full-speed PHY
- It includes full support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - It allows host to turn $V_{BUS}$ off to conserve battery power in OTG applications
  - It supports OTG monitoring of $V_{BUS}$ levels with internal comparators
  - It supports dynamic host-peripheral switch of role
- It is software-configurable to operate as:
  - USB On-The-Go Full-Speed Dual Role device
- It supports FS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 1.25 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2.
61.2.2 Host-mode features

The OTG_FS interface main features and requirements in host-mode are the following:

- External charge pump for $V_{BUS}$ voltage generation.
- Up to 12 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 12 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 12 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

61.2.3 Peripheral-mode features

The OTG_FS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 5 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 5 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 6 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.
## 61.3 OTG_FS implementation

<table>
<thead>
<tr>
<th>USB features</th>
<th>OTG_FS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device bidirectional endpoints (including EP0)</td>
<td>6</td>
</tr>
<tr>
<td>Host mode channels</td>
<td>12</td>
</tr>
<tr>
<td>Size of dedicated SRAM</td>
<td>1.2 Kbytes</td>
</tr>
<tr>
<td>USB 2.0 link power management (LPM) support</td>
<td>X</td>
</tr>
<tr>
<td>OTG revision supported</td>
<td>2.0</td>
</tr>
<tr>
<td>Battery charging detection (BCD) support</td>
<td>X</td>
</tr>
<tr>
<td>Integrated PHY</td>
<td>FS</td>
</tr>
</tbody>
</table>

1. “X” = supported, “-” = not supported, “FS” = supported in FS mode, “HS” = supported in HS mode.
61.4 OTG_FS functional description

61.4.1 OTG_FS block diagram

Figure 892. OTG_FS full-speed block diagram

61.4.2 OTG_FS pin and internal signals

Table 652. OTG_FS input/output pins

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_FS_DP</td>
<td>Digital input/output</td>
<td>USB OTG D+ line</td>
</tr>
<tr>
<td>OTG_FS_DM</td>
<td>Digital input/output</td>
<td>USB OTG D- line</td>
</tr>
<tr>
<td>OTG_FS_ID</td>
<td>Digital input</td>
<td>USB OTG ID</td>
</tr>
<tr>
<td>OTG_FS_VBUS</td>
<td>Analog input</td>
<td>USB OTG VBUS</td>
</tr>
</tbody>
</table>
61.4.3 OTG_FS core

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in Section 61.13: OTG_FS interrupts.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 1.25-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG FS peripheral, the AHB frequency should be higher than 14.2 MHz.

61.4.4 Embedded full-speed OTG PHY connected to OTG_FS

The embedded full-speed OTG PHY is controlled by the OTG FS core and conveys USB control & data signals through the full-speed subset of the UTMI+ Bus (UTMIFS). It provides the physical support to USB connectivity.

The full-speed OTG PHY includes the following components:

- FS/LS transceiver module used by both host and device. It directly drives transmission and reception on the single-ended USB lines.
- DP/DM integrated pull-up and pull-down resistors controlled by the OTG_FS core depending on the current role of the device. As a peripheral, it enables the DP pull-up resistor to signal full-speed peripheral connections as soon as $V_{BUS}$ is sensed to be at a valid level (B-session valid). In host mode, pull-down resistors are enabled on both DP/DM. Pull-up and pull-down resistors are dynamically switched when the role of the device is changed via the host negotiation protocol (HNP).
- Pull-up/pull-down resistor ECN circuit. The DP pull-up consists of two resistors controlled separately from the OTG_FS as per the resistor Engineering Change Notice applied to USB Rev2.0. The dynamic trimming of the DP pull-up strength allows for better noise rejection and Tx/Rx signal quality.

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usb_sof</td>
<td>Digital output</td>
<td>USB OTG start-of-frame event for on-chip peripherals</td>
</tr>
<tr>
<td>usb_wkup</td>
<td>Digital output</td>
<td>USB OTG wakeup event output</td>
</tr>
<tr>
<td>usb_gbl_it</td>
<td>Digital output</td>
<td>USB OTG global interrupt</td>
</tr>
</tbody>
</table>
61.4.5 OTG detections

Additionally the OTG_FS uses the following functions:
- integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- \( V_{BUS} \) sensing comparators with hysteresis used to detect \( V_{BUS} \) valid, A-B session valid and session-end voltage thresholds. They are used to detect valid startup and end-of-session conditions, and constantly monitor the \( V_{BUS} \) supply during USB operations.

61.5 OTG_FS dual role device (DRD)

Figure 893. OTG_FS A-B device connection

1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

61.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG_FS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG_FS issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG_FS complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
61.6 **OTG_FS as a USB peripheral**

This section gives the functional description of the OTG_FS in the USB peripheral mode. The OTG_FS works as an USB peripheral in the following circumstances:

- **OTG B-Peripheral**
  - OTG B-device default state if B-side of USB cable is plugged in
- **B-device**
  - If the ID line is present, functional and connected to the B-side of the USB cable.
- **Peripheral only** (see Figure 894: OTG_FS peripheral-only connection)
  - The force device mode bit (FDMOD) in the Section 61.15.4: OTG USB configuration register (OTG_GUSBCFG) is set to 1, forcing the OTG_FS core to work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

**Note:** To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from VBUS.

**Figure 894. OTG_FS peripheral-only connection**

![Diagram](MSv36916V2)

1. Use a regulator to build a bus-powered device.

### 61.6.1 Peripheral states

#### Powered state

The VBUS input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG_FS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The VBUS input also ensures that valid VBUS levels are supplied by the host during USB operations. If a drop in VBUS below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG_FS...
automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG_FS has exited the powered state.

In the powered state, the OTG_FS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG_FS enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG_FS expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG_FS then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG_FS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG_FS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wakeup signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

61.6.2 Peripheral endpoints

The OTG_FS core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Proper control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt
(OTG_DIEPINT0/OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints

- **5 IN endpoints**
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has proper control (OTG_DIEPCTLx), transfer configuration (OTG_DIEPTSIZx), and status-interrupt (OTG_DIEPINTx) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

- **5 OUT endpoints**
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a proper control (OTG_DOEPCTLx), transfer configuration (OTG_DOEPTSIZx) and status-interrupt (OTG_DOEPINTx) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

**Endpoint control**

- The following endpoint controls are available to the application through the device endpoint-x IN/OUT control register (OTG_DIEPCTLx/OTG_DOEPCTLx):
  - Endpoint enable/disable
  - Endpoint activate in current configuration
  - Program USB transfer type (isochronous, bulk, interrupt)
  - Program supported packet size
  - Program Tx FIFO number associated with the IN endpoint
  - Program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - Program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - Optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - Optionally program the STALL bit to always stall host tokens to that endpoint
  - Optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data
Endpoint transfer

The device endpoint-x transfer size registers (OTG_DIEPTSIZx/OTG_DOEPTSIZx) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG_FS core updates them with the current transfer status.

The following transfer parameters can be programmed:
- Transfer size in bytes
- Number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-x interrupt registers (OTG_DIEPINTx/OTG_DOEPINTx) indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

The peripheral core provides the following status checks and interrupt generation:
- Transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- Setup stage has been done (control-out only)
- Associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- Babble error condition has been detected
- Endpoint disable by application is effective
- Endpoint NAK by application is effective (isochronous-in only)
- More than 3 back-to-back setup packets were received (control-out only)
- Timeout condition detected (control-in only)
- Isochronous out packet has been dropped, without generating an interrupt
61.7 **OTG_FS as a USB host**

This section gives the functional description of the OTG_FS in the USB host mode. The OTG_FS works as a USB host in the following circumstances:

- **OTG A-host**
  - OTG A-device default state when the A-side of the USB cable is plugged in

- **A-device**
  - If the ID line is present, functional and connected to the A-side of the USB cable. Integrated pull-down resistors are automatically set on the DP/DM lines.

- **Host only**
  - The force host mode bit (FHMOD) in the **OTG USB configuration register (OTG_GUSBCFG)** forces the OTG_FS core to work as a USB host-only. In this case, the ID line is ignored even if present on the USB connector. Integrated pull-down resistors are automatically set on the DP/DM lines.

**Note:** On-chip 5 V VBUS generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V VBUS line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

**Figure 895. OTG_FS host-only connection**

1. VDD range is between 2 V and 3.6 V.

### 61.7.1 USB host states

**Host port power**

On-chip 5 V VBUS generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V VBUS line. The external charge pump can be driven by any GPIO output or via an I²C interface connected to an external PMIC (power management IC). When the application decides to power on VBUS, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

**VBUS valid**

In Host mode, the VBUS sensing pin does not need to be connected to VBUS.
The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the \( V_{BUS} \) generation and clear the port power bit.

**Host detection of a peripheral connection**

USB peripherals or B-device are detected as soon as they are connected. The OTG_FS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

**Host detection of peripheral a disconnection**

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

**Host enumeration**

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

**Host suspend**

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG_FS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device’s initiative (remote wakeup). In this case the remote wakeup interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wakeup signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

**61.7.2 Host channels**

The OTG_FS core instantiates 12 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 12 transfer requests at the same time. If more than 12 transfer requests are pending from the application, the host
controller driver (HCD) must re-allocate channels when they become available from 
previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic 
transaction. Each host channel makes use of proper control (OTG_HCCHARx), transfer 
configuration (OTG_HCTSIZx) and status/interrupt (OTG_HCINTx) registers with 
associated mask (OTG_HCINTMSKx) registers.

Host channel control

- The following host channel controls are available to the application through the host 
  channel-x characteristics register (OTG_HCCHARx):
  - Channel enable/disable
  - Program the FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

- The host channel-x interrupt register (OTG_HCINTx) indicates the status of an endpoint 
  with respect to USB- and AHB-related events. The application must read these register 
  when the host channels interrupt bit in the core interrupt register (HCINT bit in 
  OTG_GINTSTS) is set. Before the application can read these registers, it must first read the 
  host all channels interrupt (OTG_HAINT) register to get the exact channel number for the 
  host channel-x interrupt register. The application must clear the appropriate bit in this 
  register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.
The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

### 61.7.3 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG_FS core is fully responsible for the management of the periodic and nonperiodic request queues. The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of 8 entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of 8 pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the
PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

61.8 OTG_FS SOF trigger

Figure 896. SOF connectivity (SOF trigger output to TIM and ITR1 connection)

The OTG_FS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

61.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

61.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated. The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.
The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

### 61.9 OTG_FS low-power modes

Table 654 below defines the STM32 low power modes and their compatibility with the OTG.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>USB compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>MCU fully active</td>
<td>Required when USB not in suspend state.</td>
</tr>
<tr>
<td>Sleep</td>
<td>USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.</td>
<td>Available while USB is in suspend state.</td>
</tr>
<tr>
<td>Stop</td>
<td>USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).</td>
<td>Available while USB is in suspend state.</td>
</tr>
<tr>
<td>Standby</td>
<td>Powered-down. The peripheral must be reinitialized after exiting Standby mode.</td>
<td>Not compatible with USB applications.</td>
</tr>
</tbody>
</table>

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to Section 6: Power control (PWR) to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

The power consumption of the OTG PHY is controlled by two or three bits in the general core configuration register, depending on OTG revision supported.

- **PHY power down (OTG_GCCFG/PWRDWN)**
  - It switches on/off the full-speed transceiver module of the PHY. It must be preliminarily set to allow any USB operation

- **V\textsubscript{BUS} detection enable (OTG_GCCFG/VBDEN)**
  - It switches on/off the V\textsubscript{BUS} sensing comparators associated with OTG operations

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- **Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)**
  - When setting the stop PHY clock bit in the clock gating control register, most of the 48 MHz clock domain internal to the OTG core is switched off by clock gating. The dynamic power consumption due to the USB clock switching activity is cut even if the 48 MHz clock input is kept running by the application
  - Most of the transceiver is also disabled, and only the part in charge of detecting the asynchronous resume or remote wakeup event is kept alive.

- **Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)**
  - When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_FS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to
the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- **USB system stop**
  When the OTG_FS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG_FS core automatically reactivates both system and USB clocks by asynchronous detection of remote wakeup (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG_FS core.

### 61.10 OTG_FS Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of SOF framing period in host mode allowing to synchronize an external device with the SOF frames.

When the OTG_HFIR register is changed within a current SOF frame, the SOF period correction is applied in the next frame as described in Figure 897.

For a dynamic update, it is required to set RLDCTRL=1.

![Figure 897. Updating OTG_HFIR dynamically (RLDCTRL = 1)](ai18440b)

### 61.11 OTG_FS data FIFOs

The USB system features 1.25 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG_FS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device’s role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.
61.11.1 Peripheral FIFO architecture

Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG_FS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.
Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

61.11.2 Host FIFO architecture

Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG_FS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.
Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG_FS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG_FS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG_FS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

61.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application should allocate RAM for SETUP packets:
- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.
Device RxFIFO =

\[(5 \times \text{number of control endpoints} + 8) + (((\text{largest USB packet used} \div 4) + 1 \text{ for status information}) + (2 \times \text{number of OUT endpoints}) + 1 \text{ for Global NAK})\]

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = \((5 \times 1 + 8) + ((1,024 \div 4) + 1) + (2 \times 4) + 1 = 279\)

**Transmit FIFO RAM allocation**: the minimum RAM space required for each IN endpoint

Transmit FIFO is the maximum packet size for that particular IN endpoint.

**Note**: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

**Host mode**

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of \((\text{largest packet size} \div 4) + 1\) must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two \((\text{largest packet size} \div 4) + 1\) spaces must be allocated to receive back-to-back packets. Typically, two \((\text{largest packet size} \div 4) + 1\) spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = \((\text{largest USB packet used} \div 4) + 1 \text{ for status information} + 1 \text{ transfer complete}\)

Example: Host RxFIFO = \(((1,024 \div 4) + 1) + 1 = 258\)

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = \((512 \div 4) = 128\)

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = \((1,024 \div 4) = 256\)

**Note**: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.
61.12 OTG_FS system performance

Best USB and system performance is achieved owing to the large RAM buffers, the highly configurable FIFO sizes, the quick 32-bit FIFO access through AHB push/pop registers and, especially, the advanced FIFO control mechanism. Indeed, this mechanism allows the OTG_FS to fill in the available RAM space at best regardless of the current USB sequence. With these features:

- The application gains good margins to calibrate its intervention in order to optimize the CPU bandwidth usage:
  - It can accumulate large amounts of transmission data in advance compared to when they are effectively sent over the USB
  - It benefits of a large time margin to download data from the single receive FIFO
- The USB core is able to maintain its full operating rate, that is to provide maximum full-speed bandwidth with a great margin of autonomy versus application intervention:
  - It has a large reserve of transmission data at its disposal to autonomously manage the sending of data over the USB
  - It has a lot of empty space available in the receive buffer to autonomously fill it in with the data coming from the USB

As the OTG_FS core is able to fill in the 1.25-Kbyte RAM buffer very efficiently, and as 1.25-Kbyte of transmit/receive data is more than enough to cover a full speed frame, the USB system is able to withstand the maximum full-speed data rate for up to one USB frame (1 ms) without any CPU intervention.

61.13 OTG_FS interrupts

When the OTG_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

*Figure 900* shows the interrupt hierarchy.
OTG_FS_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

MSv36921V4
61.14 OTG_FS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_FS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG_FS registers must be accessed by words (32 bits).

CSRs are classified as follows:
- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG_FS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

61.14.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Address offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GOTGCTL</td>
<td>0x000</td>
<td>Section 61.15.1: OTG control and status register (OTG_GOTGCTL)</td>
</tr>
<tr>
<td>OTG_GOTGINT</td>
<td>0x004</td>
<td>Section 61.15.2: OTG interrupt register (OTG_GOTGINT)</td>
</tr>
<tr>
<td>OTG_GAHBCFG</td>
<td>0x008</td>
<td>Section 61.15.3: OTG AHB configuration register (OTG_GAHBCFG)</td>
</tr>
<tr>
<td>OTG_GUSBCFG</td>
<td>0x00C</td>
<td>Section 61.15.4: OTG USB configuration register (OTG_GUSBCFG)</td>
</tr>
<tr>
<td>OTG_GRSTCTL</td>
<td>0x010</td>
<td>Section 61.15.5: OTG reset register (OTG_GRSTCTL)</td>
</tr>
<tr>
<td>OTG_GINTSTS</td>
<td>0x014</td>
<td>Section 61.15.6: OTG core interrupt register [alternate] (OTG_GINTSTS)</td>
</tr>
</tbody>
</table>
<pre><code>                                                                                     | Section 61.15.7: OTG core interrupt register [alternate] (OTG_GINTSTS)         |
</code></pre>
### Table 655. Core global control and status registers (CSRs) (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Address offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GINTMSK</td>
<td>0x018</td>
<td>Section 61.15.8: OTG interrupt mask register [alternate] (OTG_GINTMSK)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 61.15.9: OTG interrupt mask register [alternate] (OTG_GINTMSK)</td>
</tr>
<tr>
<td>OTG_GRXSTSR</td>
<td>0x01C</td>
<td>Section 61.15.10: OTG receive status debug read [alternate] (OTG_GRXSTSR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 61.15.11: OTG receive status debug read register [alternate] (OTG_GRXSTSP)</td>
</tr>
<tr>
<td>OTG_GRXSTSP</td>
<td>0x020</td>
<td>Section 61.15.12: OTG status read and pop registers [alternate] (OTG_GRXSTSP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 61.15.13: OTG status read and pop registers [alternate] (OTG_GRXSTSP)</td>
</tr>
<tr>
<td>OTG_GRXFSIZ</td>
<td>0x024</td>
<td>Section 61.15.14: OTG receive FIFO size register (OTG_GRXFSIZ)</td>
</tr>
<tr>
<td>OTG_HNPTXFSIZ/OTG_DIEPTXF0(1)</td>
<td>0x028</td>
<td>Section 61.15.15: OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 61.15.16: Endpoint 0 Transmit FIFO size [alternate] (OTG_DIEPTXF0)</td>
</tr>
<tr>
<td>OTG_HNPTXSTS</td>
<td>0x02C</td>
<td>Section 61.15.17: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)</td>
</tr>
<tr>
<td>OTG_GCCFG</td>
<td>0x038</td>
<td>Section 61.15.18: OTG general core configuration register (OTG_GCCFG)</td>
</tr>
<tr>
<td>OTG_CID</td>
<td>0x03C</td>
<td>Section 61.15.19: OTG core ID register (OTG_CID)</td>
</tr>
<tr>
<td>OTG_GLPMCFG</td>
<td>0x54</td>
<td>Section 61.15.20: OTG core LPM configuration register (OTG_GLPMCFG)</td>
</tr>
<tr>
<td>OTG_HPTXFSIZ</td>
<td>0x100</td>
<td>Section 61.15.21: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)</td>
</tr>
<tr>
<td>OTG_DIEPTXFx</td>
<td>0x104 0x108 ... 0x114</td>
<td>Section 61.15.22: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)</td>
</tr>
</tbody>
</table>

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

### Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

### Table 656. Host-mode control and status registers (CSRs)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HCFCG</td>
<td>0x400</td>
<td>Section 61.15.24: OTG host configuration register (OTG_HCFCG)</td>
</tr>
<tr>
<td>OTG_HIFIC</td>
<td>0x404</td>
<td>Section 61.15.25: OTG host frame interval register (OTG_HIFIC)</td>
</tr>
<tr>
<td>OTG_HFNUM</td>
<td>0x408</td>
<td>Section 61.15.26: OTG host frame number/frame time remaining register (OTG_HFNUM)</td>
</tr>
</tbody>
</table>
Table 656. Host-mode control and status registers (CSRs) (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HPTXSTS</td>
<td>0x410</td>
<td>Section 61.15.27: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)</td>
</tr>
<tr>
<td>OTG_HAINT</td>
<td>0x414</td>
<td>Section 61.15.28: OTG host all channels interrupt register (OTG_HAINT)</td>
</tr>
<tr>
<td>OTG_HAINTMSK</td>
<td>0x418</td>
<td>Section 61.15.29: OTG host all channels interrupt mask register (OTG_HAINTMSK)</td>
</tr>
<tr>
<td>OTG_HPRT</td>
<td>0x440</td>
<td>Section 61.15.30: OTG host port control and status register (OTG_HPRT)</td>
</tr>
<tr>
<td>OTG_HCCHARx</td>
<td>0x500, 0x520, ..</td>
<td>Section 61.15.31: OTG host channel x characteristics register (OTG_HCCHARx)</td>
</tr>
<tr>
<td>OTG_HCINTx</td>
<td>0x508, 0x528, ..</td>
<td>Section 61.15.32: OTG host channel x interrupt register (OTG_HCINTx)</td>
</tr>
<tr>
<td>OTG_HCINTMSKx</td>
<td>0x50C, 0x52C, ..</td>
<td>Section 61.15.33: OTG host channel x interrupt mask register (OTG_HCINTMSKx)</td>
</tr>
<tr>
<td>OTG_HCTSIZx</td>
<td>0x510, 0x530, ..</td>
<td>Section 61.15.34: OTG host channel x transfer size register (OTG_HCTSIZx)</td>
</tr>
</tbody>
</table>

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 657. Device-mode control and status registers

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_DCFG</td>
<td>0x800</td>
<td>Section 61.15.36: OTG device configuration register (OTG_DCFG)</td>
</tr>
<tr>
<td>OTG_DCTL</td>
<td>0x804</td>
<td>Section 61.15.37: OTG device control register (OTG_DCTL)</td>
</tr>
<tr>
<td>OTG_DSTS</td>
<td>0x808</td>
<td>Section 61.15.38: OTG device status register (OTG_DSTS)</td>
</tr>
<tr>
<td>OTG_DIEPMSK</td>
<td>0x810</td>
<td>Section 61.15.39: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</td>
</tr>
<tr>
<td>OTG_DOEPMSK</td>
<td>0x814</td>
<td>Section 61.15.40: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)</td>
</tr>
<tr>
<td>OTG_DAINT</td>
<td>0x818</td>
<td>Section 61.15.41: OTG device all endpoints interrupt register (OTG_DAINT)</td>
</tr>
<tr>
<td>Acronym</td>
<td>Offset address</td>
<td>Register name</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>OTG_DAINTMSK</td>
<td>0x81C</td>
<td>Section 61.15.42: OTG all endpoints interrupt mask register (OTG_DAINTMSK)</td>
</tr>
<tr>
<td>OTG_DIEPEMPMSK</td>
<td>0x834</td>
<td>Section 61.15.43: OTG device IN endpoint FIFO empty interrupt mask register</td>
</tr>
<tr>
<td>OTG_DEACHINT</td>
<td>0x838</td>
<td>Section 61.15.44: OTG device each endpoint interrupt register (OTG_DEACHINT)</td>
</tr>
<tr>
<td>OTG_DEACHINTMSK</td>
<td>0x83C</td>
<td>Section 61.15.45: OTG device each endpoint interrupt mask register (OTG_DEACHINTMSK)</td>
</tr>
<tr>
<td>OTG_HS_DIEPEACHMSK1</td>
<td>0x844</td>
<td>Section 61.15.46: OTG device each IN endpoint-1 interrupt mask register</td>
</tr>
<tr>
<td>OTG_HS_DOEPEACHMSK1</td>
<td>0x884</td>
<td>Section 61.15.47: OTG device each OUT endpoint-1 interrupt mask register</td>
</tr>
<tr>
<td>OTG_DIEPCTL0</td>
<td>0x900</td>
<td>Section 61.15.48: OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)</td>
</tr>
<tr>
<td>OTG_DIEPCTLx</td>
<td>0x920, 0x940,</td>
<td>Section 61.15.49: OTG device IN endpoint x control register [alternate]</td>
</tr>
<tr>
<td></td>
<td>0x9A0</td>
<td>Section 61.15.50: OTG device IN endpoint x control register [alternate]</td>
</tr>
<tr>
<td>OTG_DIEPINTx</td>
<td>0x908, 0x928,</td>
<td>Section 61.15.51: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)</td>
</tr>
<tr>
<td></td>
<td>0x988</td>
<td></td>
</tr>
<tr>
<td>OTG_DIEPTSIZ0</td>
<td>0x910</td>
<td>Section 61.15.52: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)</td>
</tr>
<tr>
<td>OTG_DTXFSTSx</td>
<td>0x918, 0x938,</td>
<td>Section 61.15.53: OTG device IN endpoint transmit FIFO status register</td>
</tr>
<tr>
<td></td>
<td>0x998</td>
<td>(OTG_DTXFSTSx)</td>
</tr>
<tr>
<td>OTG_DIEPTSIZx</td>
<td>0x930, 0x950,</td>
<td>Section 61.15.54: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)</td>
</tr>
<tr>
<td></td>
<td>0x980</td>
<td></td>
</tr>
<tr>
<td>OTG_DOEPCTL0</td>
<td>0xB00</td>
<td>Section 61.15.55: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)</td>
</tr>
<tr>
<td>OTG_DOEPIINTx</td>
<td>0xB08, 0xB28,</td>
<td>Section 61.15.56: OTG device OUT endpoint x interrupt register (OTG_DOEPIINTx)</td>
</tr>
<tr>
<td></td>
<td>0xBA8</td>
<td></td>
</tr>
</tbody>
</table>
Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

### Table 658. Data FIFO (DFIFO) access register map

<table>
<thead>
<tr>
<th>FIFO access register section</th>
<th>Offset address</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device IN endpoint 0/Host OUT Channel 0: DFIFO write access</td>
<td>0x1000–0x1FFC</td>
<td>w</td>
</tr>
<tr>
<td>Device OUT endpoint 0/Host IN Channel 0: DFIFO read access</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>Device IN endpoint 1/Host OUT Channel 1: DFIFO write access</td>
<td>0x2000–0x2FFC</td>
<td>w</td>
</tr>
<tr>
<td>Device OUT endpoint 1/Host IN Channel 1: DFIFO read access</td>
<td></td>
<td>r</td>
</tr>
</tbody>
</table>

... ...

| Device IN endpoint x(1)/Host OUT Channel x(1): DFIFO write access | 0xX000–0xXFFC | w      |
| Device OUT endpoint x(1)/Host IN Channel x(1): DFIFO read access |             | r      |

1. Where x is 5 in device mode and 11 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

### Table 659. Power and clock gating control and status registers

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_PCGCCTL</td>
<td>0xE00–0xE04</td>
<td>Section 61.15.61: OTG power and clock gating control register (OTG_PCGCCTL)</td>
</tr>
</tbody>
</table>
## 61.15 OTG_FS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

### 61.15.1 OTG control and status register (OTG_GOTGCTL)

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

**Address offset:** 0x000  
**Reset value:** 0x0001 0000

<table>
<thead>
<tr>
<th>Bit 31-22</th>
<th>Description</th>
</tr>
</thead>
</table>
| CURMOD | Current mode of operation  
Indicates the current mode (host or device).  
0: Device mode  
1: Host mode |
| OTGVER | OTG version  
Selects the OTG revision.  
0: OTG Version 1.3. OTG1.3 is obsolete for new product development.  
1: OTG Version 2.0. In this version the core supports only data line pulsing for SRP. |
| BSVLD | B-session valid  
Indicates the device mode transceiver status.  
0: B-session is not valid.  
1: B-session is valid.  
In OTG mode, the user can use this bit to determine if the device is connected or disconnected.  
*Note: Only accessible in device mode.* |
| ASVLD | A-session valid  
Indicates the host mode transceiver status.  
0: A-session is not valid  
1: A-session is valid  
*Note: Only accessible in host mode.* |
| DBCT | Long/short debounce time  
Indicates the debounce time of a detected connection.  
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)  
1: Short debounce time, used for soft connections (2.5 µs)  
*Note: Only accessible in host mode.* |

Bits 31:22  
Reserved, must be kept at reset value.
Bit 16 **CIDSTS**: Connector ID status
Indicates the connector ID status on a connect event.
0: The OTG_FS controller is in A-device mode
1: The OTG_FS controller is in B-device mode
*Note:* Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN**: Embedded host enable
It is used to select between OTG A device state machine and embedded host state machine.
0: OTG A device state machine is selected
1: Embedded host state machine is selected

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **BVALOVAL**: B-peripheral session valid override value.
This bit is used to set override value for Bvalid signal when BVALOEN bit is set.
0: Bvalid value is '0' when BVALOEN = 1
1: Bvalid value is '1' when BVALOEN = 1
*Note:* Only accessible in device mode.

Bit 6 **BVALOEN**: B-peripheral session valid override enable.
This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.
0: Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core
1: Internally Bvalid received from the PHY is overridden with BVALOVAL bit value
*Note:* Only accessible in device mode.

Bit 5 **AVALOVAL**: A-peripheral session valid override value.
This bit is used to set override value for Avalid signal when AVALOEN bit is set.
0: Avalid value is '0' when AVALOEN = 1
1: Avalid value is '1' when AVALOEN = 1
*Note:* Only accessible in host mode.

Bit 4 **AVALOEN**: A-peripheral session valid override enable.
This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.
0: Override is disabled and Avalid signal from the respective PHY selected is used internally by the core
1: Internally Avalid received from the PHY is overridden with AVALOVAL bit value
*Note:* Only accessible in host mode.
61.15.2 OTG interrupt register (OTG_GOTGINT)

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 ADTOCHG: A-device timeout change
The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both device and host modes.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 SEDET: Session end detected
The core sets this bit to indicate that the level of the voltage on VBUS is no longer valid for a B-Peripheral session when VBUS < 0.8 V.

Note: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.
61.15.3 OTG AHB configuration register (OTG_GAHBCFG)

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:9</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 8</td>
<td>PTXFE LVL: Periodic Tx FIFO empty level</td>
</tr>
<tr>
<td></td>
<td>Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered:</td>
</tr>
<tr>
<td></td>
<td>0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty</td>
</tr>
<tr>
<td></td>
<td>1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty</td>
</tr>
<tr>
<td>Note:</td>
<td>Only accessible in host mode.</td>
</tr>
<tr>
<td>Bit 7</td>
<td>TXFE LVL: Tx FIFO empty level</td>
</tr>
<tr>
<td>Condition: device mode. This bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:</td>
<td></td>
</tr>
<tr>
<td>0: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty</td>
<td></td>
</tr>
<tr>
<td>1: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty</td>
<td></td>
</tr>
<tr>
<td>Condition: host mode. This bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:</td>
<td></td>
</tr>
<tr>
<td>0: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty</td>
<td></td>
</tr>
<tr>
<td>1: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 6:1 | Reserved, must be kept at reset value. |
| Bit 0   | GINTMSK: Global interrupt mask |
| The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit’s setting, the interrupt status registers are updated by the core. |
| 0: Mask the interrupt assertion to the application. |
| 1: Unmask the interrupt assertion to the application. |
| Note: Accessible in both device and host modes. |
61.15.4 OTG USB configuration register (OTG_GUSBCFG)

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Address offset: 0x00C
Reset value: 0x0000 1440

<table>
<thead>
<tr>
<th>Address Offset: 0x00C</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bit 30  **FDMOD**: Force device mode
  Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
  0: Normal mode
  1: Force device mode
  After setting the force bit, the application must wait at least 25 ms before the change takes effect.

  *Note:* Accessible in both device and host modes.

Bit 29  **FHMOD**: Force host mode
  Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
  0: Normal mode
  1: Force host mode
  After setting the force bit, the application must wait at least 25 ms before the change takes effect.

  *Note:* Accessible in both device and host modes.

Bits 28:14  Reserved, must be kept at reset value.

Bits 13:10  **TRDT[3:0]**: USB turnaround time
  These bits allows to set the turnaround time in PHY clocks. They must be configured according to Table 660: TRDT values, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

  *Note:* Only accessible in device mode.

Bits 9:7  Reserved, must be kept at reset value.
Bit 6 **PHYSEL**: Full Speed serial transceiver mode select
This bit is always 1 with read-only access.

Bits 5:3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]**: FS timeout calibration
The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.
The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

<table>
<thead>
<tr>
<th>AHB frequency range (MHz)</th>
<th>TRDT minimum value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Min</strong></td>
<td><strong>Max</strong></td>
</tr>
<tr>
<td>14.2</td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>17.2</td>
</tr>
<tr>
<td>17.2</td>
<td>18.5</td>
</tr>
<tr>
<td>18.5</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>21.8</td>
</tr>
<tr>
<td>21.8</td>
<td>24</td>
</tr>
<tr>
<td>24</td>
<td>27.5</td>
</tr>
<tr>
<td>27.5</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>-</td>
</tr>
</tbody>
</table>

### 61.15.5 OTG reset register (OTG_GRSTCTL)

The application uses this register to reset various hardware features inside the core.

Address offset: 0x010
Reset value: 0x8000 0000
Bit 31 **AHBIDL**: AHB master idle
Indicates that the AHB master state machine is in the Idle condition.
*Note: Accessible in both device and host modes.*

Bits 30:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]**: Tx FIFO number
This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.
Condition: host mode
00000: Non-periodic Tx FIFO flush
00001: Periodic Tx FIFO flush
10000: Flush all the transmit FIFOs
Condition: device mode
00000: Tx FIFO 0 flush
00001: Tx FIFO 1 flush
00010: Tx FIFO 2 flush
...  
01111: Tx FIFO 15 flush
10000: Flush all the transmit FIFOs
*Note: Accessible in both device and host modes.*

Bit 5 **TXFFLSH**: Tx FIFO flush
This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.
The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:
Read—NAK Effective interrupt ensures the core is not reading from the FIFO
Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.
Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.
*Note: Accessible in both device and host modes.*

Bit 4 **RXFFLSH**: Rx FIFO flush
The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.
The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.
The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.
*Note: Accessible in both device and host modes.*

Bit 3 Reserved, must be kept at reset value.
Bit 2  **FCRST**: Host frame counter reset  
The application writes this bit to reset the frame number counter inside the core. When the frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0. When application writes “1” to the bit, it might not be able to read back the value as it gets cleared by the core in a few clock cycles.  
*Note: Only accessible in host mode.*

Bit 1  **PSRST**: Partial soft reset  
Resets the internal state machines but keeps the enumeration info. Could be used to recover some specific PHY errors.  
*Note: Accessible in both device and host modes.*

Bit 0  **CSRST**: Core soft reset  
Resets the HCLK and PHY clock domains as follows:  
Cleans the interrupts and all the CSR register bits except for the following bits:  
- GATEHCLK bit in OTG_PCGCCTL  
- STPPCLK bit in OTG_PCGCCTL  
- FSLSPCS bits in OTG_HCFG  
- DSPD bit in OTG_DCFG  
- SDIS bit in OTG_DCTL  
- OTG_GCCFG register  
- FDMOD bit in OTG_GUSBCFG  
- FHMOD bit in OTG_GUSBCFG  
All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.  
Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.  
Typically, the software reset is used during software development and also when the user dynamically changes the PHY selection bits in the above listed USB configuration registers. When the user changes the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.  
*Note: Accessible in both device and host modes.*
### 61.15.6 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Host mode, see next section for Device mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>WKUP</th>
<th>INT</th>
<th>Bit 30</th>
<th>SRQ</th>
<th>INT</th>
<th>Bit 29</th>
<th>DISC</th>
<th>INT</th>
<th>Bit 28</th>
<th>CIDS</th>
<th>CHG</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

| Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|----|----|----|----|----|---|---|---|---|---|---|---|---|---|---|---|
| EOFP | ISO0 | DRP | ENUM | DNE | USB | RST | USB | SUSP | Res. | Res. | GO NAK | EFF | GI NAK | EFF | NPTXF | E | RXF | LVL | SOF | OTG INT | MMIS | CMOD |
| rc_w1 | rc_w1 | rc_w1 | rc_w1 | rc_w1 | r | r | r | r | r | rc_w1 | r | rc_w1 |

**Bit 31 WKUPINT:** Resume/remote wakeup detected interrupt

Wakeup interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

  In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.

– During LPM(L1):

  This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.

*Note:* Accessible in both device and host modes.

**Bit 30 SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when VBUS is in the valid range for a B-peripheral device. Accessible in both device and host modes.

**Bit 29 DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

*Note:* Only accessible in host mode.

**Bit 28 CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

*Note:* Accessible in both device and host modes.
Bit 27  **LPMINT**: LPM interrupt
In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.
In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).
This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26  **PTXFE**: Periodic Tx FIFO empty
Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

*Note: Only accessible in host mode.*

Bit 25  **HCINT**: Host channels interrupt
The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

*Note: Only accessible in host mode.*

Bit 24  **HPRTINT**: Host port interrupt
The core sets this bit to indicate a change in port status of one of the OTG_FS controller ports in host mode. The application must read the OTG_HPR register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPR register to clear this bit.

*Note: Only accessible in host mode.*

Bit 23  **RSTDET**: Reset detected interrupt
In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

*Note: Only accessible in device mode.*

Bit 22  Reserved, must be kept at reset value.

Bit 21  **IPXFR**: Incomplete periodic transfer
In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

Bit 20  **ISOIXFR**: Incomplete isochronous IN transfer
The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

*Note: Only accessible in device mode.*

Bit 19  **OEPINT**: OUT endpoint interrupt
The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

*Note: Only accessible in device mode.*
Bit 18  **IEPINT**: IN endpoint interrupt
The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.
*Note: Only accessible in device mode.*

Bits 17:16  Reserved, must be kept at reset value.

Bit 15  **EOPf**: End of periodic frame interrupt
Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.
*Note: Only accessible in device mode.*

Bit 14  **ISOODRP**: Isochronous OUT packet dropped interrupt
The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.
*Note: Only accessible in device mode.*

Bit 13  **ENUMDNE**: Enumeration done
The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.
*Note: Only accessible in device mode.*

Bit 12  **USBRST**: USB reset
The core sets this bit to indicate that a reset is detected on the USB.
*Note: Only accessible in device mode.*

Bit 11  **USBSUSP**: USB suspend
The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.
*Note: Only accessible in device mode.*

Bit 10  **ESUSP**: Early suspend
The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.
*Note: Only accessible in device mode.*

Bits 9:8  Reserved, must be kept at reset value.

Bit 7  **GONAKEFF**: Global OUT NAK effective
Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).
*Note: Only accessible in device mode.*
Bit 6  **GINA\_EFF:** Global IN non-periodic NAK effective  
Indicates that the Set global non-periodic IN NAK bit in the OTG\_DCTL register (SGINAK bit in OTG\_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG\_DCTL register (CGINAK bit in OTG\_DCTL).  
This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.  
*Note: Only accessible in device mode.*

Bit 5  **NPT\_XFE:** Non-periodic Tx FIFO empty  
This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG\_GAHB\_CFG register (TXFELVL bit in OTG\_GAHB\_CFG).  
*Note: Accessible in host mode only.*

Bit 4  **RX\_FLVL:** Rx FIFO non-empty  
Indicates that there is at least one packet pending to be read from the Rx FIFO.  
*Note: Accessible in both host and device modes.*

Bit 3  **SOF:** Start of frame  
In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.  
In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG\_DSTS register to get the current frame number.  
This interrupt is seen only when the core is operating in FS.  
*Note: This register may return ‘1’ if read immediately after power on reset. If the register bit reads ‘1’ immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.*  
*Note: Accessible in both host and device modes.*

Bit 2  **OTG\_INT:** OTG interrupt  
The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG\_GOTG\_INT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG\_GOTG\_INT register to clear this bit.  
*Note: Accessible in both host and device modes.*

Bit 1  **MMIS:** Mode mismatch interrupt  
The core sets this bit when the application is trying to access:  
– A host mode register, when the core is operating in device mode  
– A device mode register, when the core is operating in host mode  
The register access is completed on the AHB with an OK\_AY response, but is ignored by the core internally and does not affect the operation of the core.  
*Note: Accessible in both host and device modes.*

Bit 0  **CMOD:** Current mode of operation  
Indicates the current mode.  
0: Device mode  
1: Host mode  
*Note: Accessible in both host and device modes.*
61.15.7 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Device mode, see previous section for Host mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014
Reset value: 0x0400 0020

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>WKUPINT</th>
<th>Resume/remote wakeup detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Wakeup interrupt during suspend(L2) or LPM(L1) state.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– During suspend(L2):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wakeup is detected on the USB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– During LPM(L1):</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This interrupt is asserted for either host initiated resume or device initiated remote wakeup on USB.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>SRQINT</th>
<th>Session request/new session detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when VBUS is in the valid range for a B-peripheral device. Accessible in both device and host modes.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>DISCINT</th>
<th>Disconnect detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Asserted when a device disconnect is detected.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: Only accessible in host mode.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>CIDSCGH</th>
<th>Connector ID status change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The core sets this bit when there is a change in connector ID status.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Note: Accessible in both device and host modes.</td>
<td></td>
</tr>
</tbody>
</table>
Bit 27 **LPMINT**: LPM interrupt
In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.
In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG). This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE**: Periodic Tx FIFO empty
Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).
Note: Only accessible in host mode.

Bit 25 **HCINT**: Host channels interrupt
The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.
Note: Only accessible in host mode.

Bit 24 **HPRTINT**: Host port interrupt
The core sets this bit to indicate a change in port status of one of the OTG_FS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.
Note: Only accessible in host mode.

Bit 23 **RSTDET**: Reset detected interrupt
In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.
Note: Only accessible in device mode.

Bit 22 **Reserved, must be kept at reset value.**

Bit 21 **INCOMPISOOUT**: Incomplete isochronous OUT transfer
In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **ISIOXFR**: Incomplete isochronous IN transfer
The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.
Note: Only accessible in device mode.

Bit 19 **OEPINT**: OUT endpoint interrupt
The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.
Note: Only accessible in device mode.
Bit 18  **IEPINT**: IN endpoint interrupt
The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

*Note: Only accessible in device mode.*

Bits 17:16 Reserved, must be kept at reset value.

Bit 15  **EOPF**: End of periodic frame interrupt
Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

*Note: Only accessible in device mode.*

Bit 14  **ISOODRP**: Isochronous OUT packet dropped interrupt
The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

*Note: Only accessible in device mode.*

Bit 13  **ENUMDNE**: Enumeration done
The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

*Note: Only accessible in device mode.*

Bit 12  **USBRST**: USB reset
The core sets this bit to indicate that a reset is detected on the USB.

*Note: Only accessible in device mode.*

Bit 11  **USBSUSP**: USB suspend
The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

*Note: Only accessible in device mode.*

Bit 10  **ESUSP**: Early suspend
The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

*Note: Only accessible in device mode.*

Bits 9:8 Reserved, must be kept at reset value.

Bit 7  **GONAKEFF**: Global OUT NAK effective
Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

*Note: Only accessible in device mode.*
Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL).

This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

*Note:* Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

*Note:* Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

*Note:* Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

*Note:* This register may return ‘1’ if read immediately after power on reset. If the register bit reads ‘1’ immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

*Note:* Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

*Note:* Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode
– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

*Note:* Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode
1: Host mode

*Note:* Accessible in both host and device modes.
61.15.8 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Host mode, see next section for Device mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
<th>Masking</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>WUIM</td>
<td>Resume/remote wakeup detected interrupt mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>30</td>
<td>SRQIM</td>
<td>Session request/new session detected interrupt mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>29</td>
<td>DISCINT</td>
<td>Disconnect detected interrupt mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>28</td>
<td>CIDSCHGIM</td>
<td>Connector ID status change mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>27</td>
<td>LPMINTM</td>
<td>LPM interrupt mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>26</td>
<td>PTXFEM</td>
<td>Periodic Tx FIFO empty mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>25</td>
<td>HCIM</td>
<td>Host channels interrupt mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>24</td>
<td>PRTIM</td>
<td>Host port interrupt mask</td>
<td>0: Masked interrupt, 1: Unmasked interrupt</td>
</tr>
<tr>
<td>23:22 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 21 IPXFRM: Incomplete periodic transfer mask
0: Masked interrupt, 1: Unmasked interrupt
61.15.9 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Device mode, see previous section for Host mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Masked Interrupt</th>
<th>Unmasked Interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>WUIM: Resume/remote wakeup detected</td>
<td>0: Masked</td>
<td>1: Unmasked</td>
</tr>
<tr>
<td></td>
<td>interrupt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>SRQIM: Session request/new session</td>
<td>0: Masked</td>
<td>1: Unmasked</td>
</tr>
<tr>
<td></td>
<td>detected interrupt</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Accessible in both host and device modes.
Bit 28 **CIDSCGHGM**: Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt

*Note: Accessible in both host and device modes.*

Bit 27 **LPMINTM**: LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt

*Note: Accessible in both host and device modes.*

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **RSTDETM**: Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 22 Reserved, must be kept at reset value.

Bit 21 **ISOOXFRM**: Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 20 **ISOIXFRM**: Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 19 **OEPINT**: OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 18 **IEPINT**: IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM**: End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 14 **ISOODRPM**: Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **ENUMDNEM**: Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **USBRSST**: USB reset mask
0: Masked interrupt
1: Unmasked interrupt

Bit 11 **USBSUSPM**: USB suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bit 10 **ESUSPM**: Early suspend mask
0: Masked interrupt
1: Unmasked interrupt
Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM**: Global OUT NAK effective mask
- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 6 **GINAKEFFM**: Global non-periodic IN NAK effective mask
- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **RXFLVLVM**: Receive FIFO non-empty mask
- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 3 **SOFM**: Start of frame mask
- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 2 **OTGINT**: OTG interrupt mask
- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 1 **MMISM**: Mode mismatch interrupt mask
- 0: Masked interrupt
- 1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

### 61.15.10 OTG receive status debug read [alternate] (OTG_GRXSTSR)

Valid for Host mode, see next section for Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKTSTS[3:0]</td>
<td>DPID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPID</td>
<td>BCNT[10:0]</td>
<td>CHNUM[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

2942/3791 RM0477 Rev 6
61.15.11 OTG receive status debug read register [alternate]
(OTG_GRXSTSR)

Valid for Device mode, see previous section for Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

| Bit 31:28 Reserved, must be kept at reset value. |
| Bit 27 **STSPHST**: Status phase start |
| Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern. |
| Bit 26:25 Reserved, must be kept at reset value. |
| Bit 24:21 **FRMNUM[3:0]**: Frame number |
| This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported. |
Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
- 0001: Global OUT NAK (triggers an interrupt)
- 0010: OUT data packet received
- 0011: OUT transfer completed (triggers an interrupt)
- 0100: SETUP transaction completed (triggers an interrupt)
- 0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received OUT data packet
- 00: DATA0
- 10: DATA1

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number
Indicates the endpoint number to which the current received packet belongs.

### 61.15.12 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Valid for Host mode, see next section for Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>1 1 1 1</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

[Table showing bits distribution]
Bits 31:21  Reserved, must be kept at reset value.

Bits 20:17  **PKTSTS[3:0]: Packet status**
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15  **DPID[1:0]: Data PID**
Indicates the data PID of the received packet
00: DATA0
10: DATA1

Bits 14:4  **BCNT[10:0]: Byte count**
Indicates the byte count of the received IN data packet.

Bits 3:0  **CHNUM[3:0]: Channel number**
Indicates the channel number to which the current received packet belongs.

### 61.15.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

Valid for Device mode, see previous section for Host mode.

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DPID[0]** | **BCNT[10:0]** | **EPNUM[3:0]**
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:28  Reserved, must be kept at reset value.

**Bit 27  STSPHST**: Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25  Reserved, must be kept at reset value.
61.15.14 OTG receive FIFO size register (OTG_GRXFSIZ)

The application can program the RAM size that must be allocated to the Rx FIFO.

Address offset: 0x024

Reset value: 0x0000 0200

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 RXFD[15:0]: Rx FIFO depth

This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.
61.15.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)

Valid for Host mode, see next section for Device mode.
Address offset: 0x028
Reset value: 0x0200 0200

Bits 31:16 NPTXFD[15:0]: Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 NPTXFSA[15:0]: Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

61.15.16 Endpoint 0 Transmit FIFO size [alternate] (OTG_DIEPTXF0)

Valid for Device mode, see previous section for Host mode.
Address offset: 0x028
Reset value: 0x0200 0200

Bits 31:16 TX0FD[15:0]: Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 TX0FSA[15:0]: Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.
61.15.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Host mode only, this register is not valid in Device mode.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

Address offset: 0x02C

Reset value: 0x0008 0200

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30:24</td>
<td>NPTXQTOP[6:0]: Top of the non-periodic transmit request queue</td>
</tr>
<tr>
<td>23:16</td>
<td>NPTQXSAV[7:0]: Non-periodic transmit request queue space available</td>
</tr>
<tr>
<td>15:0</td>
<td>NPTXFSAV[15:0]: Non-periodic Tx FIFO space available</td>
</tr>
</tbody>
</table>

Bit 31: Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]**: Top of the non-periodic transmit request queue
- Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
- Bits 30:27: Channel/endpoint number
- Bits 26:25:
  - XXXX00X: IN/OUT token
  - XXXX01X: Zero-length transmit packet (device IN/host OUT)
  - XXXX11X: Channel halt command
- Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]**: Non-periodic transmit request queue space available
- Indicates the amount of free space available in the non-periodic transmit request queue.
- This queue holds both IN and OUT requests.
  - 0: Non-periodic transmit request queue is full
  - 1: 1 location available
  - 2: locations available
  - n: n locations available (0 ≤ n ≤ 8)
- Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]**: Non-periodic Tx FIFO space available
- Indicates the amount of free space available in the non-periodic Tx FIFO.
- Values are in terms of 32-bit words.
  - 0: Non-periodic Tx FIFO is full
  - 1: 1 word available
  - 2: 2 words available
  - n: n words available (where 0 ≤ n ≤ 512)
- Others: Reserved
### 61.15.18 OTG general core configuration register (OTGGCCFG)

Address offset: 0x038

Reset value: 0x0000 XXXX

| Bit 31:26 | Reserved, must be kept at reset value. |
| Bit 25   | Reserved, must be kept at reset value. |
| Bit 24   | Reserved, must be kept at reset value. |
| Bit 23:22 | Reserved, must be kept at reset value. |

| Bit 21 | VBDEN: USB VBUS detection enable |
|        | Enables VBUS sensing comparators to detect VBUS valid levels on the VBUS PAD for USB host and device operation. If HNP and/or SRP support is enabled, VBUS comparators are automatically enabled independently of VBDEN value. |
|        | 0 = VBUS detection disabled |
|        | 1 = VBUS detection enabled |

| Bit 20 | SDEN: Secondary detection (SD) mode enable |
|        | This bit is set by the software to put the BCD into SD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly |

| Bit 19 | PDEN: Primary detection (PD) mode enable |
|        | This bit is set by the software to put the BCD into PD mode. Only one detection mode (DCD, PD, SD or OFF) should be selected to work correctly |

| Bit 18 | Reserved, must be kept at reset value. |

| Bit 17 | BCDEN: Battery charging detector (BCD) enable |
|        | This bit is set by the software to enable the BCD support within the USB device. When enabled, the USB PHY is fully controlled by BCD and cannot be used for normal communication. Once the BCD discovery is finished, the BCD should be placed in OFF mode by clearing this bit to ‘0’ in order to allow the normal USB operation. |

| Bit 16 | PWRDWN: Power down control of FS PHY |
|        | Used to activate the FS PHY in transmission/reception. When reset, the PHY is kept in power-down. When set, the BCD function must be off (BCDEN=0). |
|        | 0 = USB FS PHY disabled |
|        | 1 = USB FS PHY enabled |

| Bits 15:5 | Reserved, must be kept at reset value. |

| Bit 4 | SESSVLD: VBUS session indicator |
|       | Indicates if VBUS is above VBUS session threshold. |
|       | 0: VBUS is below VBUS session threshold |
|       | 1: VBUS is above VBUS session threshold |
Bit 3 **PS2DET**: DM pull-up detection status
This bit is active only during PD and gives the result of comparison between DM voltage level and VLGC threshold. In normal situation, the DM level should be below this threshold. If it is above, it means that the DM is externally pulled high. This can be caused by connection to a PS2 port (which pulls-up both DP and DM lines) or to some proprietary charger not following the BCD specification.
0: Normal port detected (connected to SDP, CDP or DCP)
1: PS2 port or proprietary charger detected

Bit 2 **SDET**: Secondary detection (SD) status
This bit gives the result of SD.
0: CD P detected
1: DCP detected

Bit 1 **PDET**: Primary detection (PD) status
This bit gives the result of PD.
0: no BCD support detected (connected to SDP or proprietary device).
1: BCD support detected (connected to CDP or DCP).

Bit 0 Reserved, must be kept at reset value.

### OTG core ID register (OTG_CID)
This is a register containing the Product ID as reset value.

Address offset: 0x03C
Reset value: 0x0000 4000

```
<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bit 15-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRODUCT_ID[31:16]</td>
<td>PRODUCT_ID[15:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>
```

Bits 31:0 **PRODUCT_ID[31:0]**: Product ID field
Application-programmable ID field.

### OTG core LPM configuration register (OTG_GLPMCFG)
Address offset: 0x054
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bit 15-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw r r r r</td>
<td>rs rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>SLP STS LPMRSP[1:0] L1DS EN BESLTHRS[3:0] L1SS EN REM WAKE BESL[3:0] LPM ACK LPM EN</td>
<td></td>
</tr>
<tr>
<td>r r r rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>
```
Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL**: Enable best effort service latency
This bit enables the BESL feature as defined in the LPM errata:

- 0: The core works as described in the following document:
  USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- 1: The core works as described in the LPM Errata:
  Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007

*Note:* Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit should be set to ‘1’ by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]**: LPM retry count status
Number of LPM host retries still remaining to be transmitted for the current LPM sequence.

*Note:* Accessible only in host mode.

Bit 24 **SNDLPM**: Send LPM transaction
When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

*Note:* This bit must be set only when the host is connected to a local port.
*Note:* Accessible only in host mode.

Bits 23:21 **LPMRCNT[2:0]**: LPM retry count
When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

*Note:* Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]**: LPM Channel Index
The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

*Note:* Accessible only in host mode.

Bit 16 **L1RSMOK**: Sleep state resume OK
Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 μs ($T_{L1Residency}$). This bit is reset when SLPSTS is 0.

- 1: The application or host can start resume from Sleep state
- 0: The application or host cannot start resume from Sleep state
Bit 15  **SLPSTS**: Port sleep status

**Device mode:**
This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the \( T_{L1TokenRetry} \) timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal. The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.
The core comes out of sleep:
– When there is any activity on the USB linestate
– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

**Host mode:**
The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.
The core clears this bit after:
– The core detects a remote L1 wakeup signal,
– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or
– The application sets the L1Resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).

0: Core not in L1
1: Core in L1

Bits 14:13  **LPMRSP[1:0]**: LPM response

**Device mode:**
The response of the core to LPM transaction received is reflected in these two bits.

**Host mode:**
Handshake response received from local device for LPM transaction
11: ACK
10: NYET
01: STALL
00: ERROR (No handshake response)

Bit 12  **L1DSEN**: L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to ‘1’ by application SW in all the cases.
Bits 11:8 **BESLTHRS[3:0]**: BESL threshold

**Device mode:**
The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

**Host mode:**
The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.

Thres[3:0] host mode resume signaling time (μs):

- 0000: 75
- 0001: 100
- 0010: 150
- 0011: 250
- 0100: 350
- 0101: 450
- 0110: 950
- All other values: reserved

Bit 7 **L1SSEN**: L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit should be set to ‘1’ by application SW in all the cases.

Bit 6 **REMWAKE**: bRemoteWake value

**Host mode:**
The value of remote wake up to be sent in the wIndex field of LPM transaction.

**Device mode (read-only):**
This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.
Bits 5:2 **BESL[3:0]:** Best effort service latency

**Host mode:**
The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{\text{LHubDrvResume1}}$ for host initiated resume.

**Device mode (read-only):**
This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

*BESL[3:0] → T_{BESL} (\mu s)*

<table>
<thead>
<tr>
<th>Value</th>
<th>Time (\mu s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>125</td>
</tr>
<tr>
<td>0001</td>
<td>150</td>
</tr>
<tr>
<td>0010</td>
<td>200</td>
</tr>
<tr>
<td>0011</td>
<td>300</td>
</tr>
<tr>
<td>0100</td>
<td>400</td>
</tr>
<tr>
<td>0101</td>
<td>500</td>
</tr>
<tr>
<td>0110</td>
<td>1000</td>
</tr>
<tr>
<td>0111</td>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
<td>3000</td>
</tr>
<tr>
<td>1001</td>
<td>4000</td>
</tr>
<tr>
<td>1010</td>
<td>5000</td>
</tr>
<tr>
<td>1011</td>
<td>6000</td>
</tr>
<tr>
<td>1100</td>
<td>7000</td>
</tr>
<tr>
<td>1101</td>
<td>8000</td>
</tr>
<tr>
<td>1110</td>
<td>9000</td>
</tr>
<tr>
<td>1111</td>
<td>10000</td>
</tr>
</tbody>
</table>

**Bit 1** **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

- **1:** ACK
  - Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:
    - No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
    - Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
    - No data pending in transmit queue (else NYET).
  - **0:** NYET
  - The preprogrammed software bit is over-ridden for response to LPM token when:
    - The received bLinkState is not L1 (STALL response), or
    - An error is detected in either of the LPM token packets because of corruption (ERROR response).

*Note:* Accessible only in device mode.

**Bit 0** **LPMEN:** LPM support enable

The application uses this bit to control the OTG_FS core LPM capabilities.

- If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.
- If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

- **0:** LPM capability is not enabled
- **1:** LPM capability is enabled
61.15.21 OTG host periodic transmit FIFO size register
(OTG_HPTXFSIZ)

Address offset: 0x100
Reset value: 0x0200 0400

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PTXFSIZ[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 **PTXFSIZ[15:0]**: Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]**: Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

61.15.22 OTG device IN endpoint transmit FIFO x size register
(OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 5)
Reset value: 0x0200 0400, 0x0200 0600, 0x0200 0800, 0x0200 0A00, 0x0200 0C00

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INEPTXFD[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INEPTXSA[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 **INEPTXFD[15:0]**: IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]**: IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

61.15.23 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:
61.15.24 OTG host configuration register (OTG_HCFG)

This register configures the core after power-on. Do not make changes to this register after initializing the host.

Address offset: 0x400
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 2 FSLSS: FS- and LS-only support
The application uses this bit to control the core’s enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.
1: FS/LS-only, even if the connected device can support HS (read-only).

Bits 1:0 FSLSPCS[1:0]: FS/LS PHY clock select

- **Condition:** FS Host mode
  - 01: PHY clock is running at 48 MHz
  - Others: Reserved

- **Condition:** LS Host mode
  - 00: Reserved
  - 01: Select 48 MHz PHY clock frequency
  - 10: Select 6 MHz PHY clock frequency
  - 11: Reserved

_Note:_ The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

61.15.25 OTG host frame interval register (OTG_HFIR)

This register stores the frame interval information for the current speed to which the OTG_FS controller has enumerated.

Address offset: 0x404
Reset value: 0x0000 EA60

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTRL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FRIVL[15:0] rw rw rw rw rw rw rw rw rw rw rw rw rw rw
### OTG host frame number/frame time remaining register (OTG_HNUM)

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

- **Address offset:** 0x408
- **Reset value:** 0x0000 3FFF

<table>
<thead>
<tr>
<th>Bit 31:17</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 16</td>
<td><strong>RLDCTRL:</strong> Reload control</td>
</tr>
<tr>
<td></td>
<td>This bit allows dynamic reloading of the HFIR register during run time.</td>
</tr>
<tr>
<td></td>
<td>0: The HFIR cannot be reloaded dynamically</td>
</tr>
<tr>
<td></td>
<td>1: The HFIR can be dynamically reloaded during run time.</td>
</tr>
<tr>
<td></td>
<td>This bit needs to be programmed during initial configuration and its value must not be changed during run time.</td>
</tr>
<tr>
<td></td>
<td><strong>Caution:</strong> RLDCTRL = 0 is not recommended.</td>
</tr>
</tbody>
</table>

- **Bits 15:0** **FRIVL[15:0]:** Frame interval

  - The value that the application programs to this field, specifies the interval between two consecutive SOFs (FS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

  \[ \text{Frame interval} = 1 \text{ ms} \times (\text{FRIVL} - 1) \]

<table>
<thead>
<tr>
<th>Bit 31:17</th>
<th><strong>FTREM[15:0]:</strong> Frame time remaining</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th><strong>FRNUM[15:0]:</strong> Frame number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.</td>
</tr>
</tbody>
</table>
61.15.27 **OTG_Host periodic transmit FIFO/queue status register**  
**(OTG_HPTXSTS)**

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

Address offset: 0x410

Reset value: 0x0008 0100

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTXQTOP[7:0]</td>
<td></td>
<td>PTXQSAV[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTXFSAVL[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 **PTXQTOP[7:0]**: Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame
0XXXXXXX: send in even frame
1XXXXXXX: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type
XXXXX00X: IN/OUT
XXXXX01X: Zero-length packet
XXXXX11X: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]**: Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic request queue. This queue holds both IN and OUT requests.

00: Periodic transmit request queue is full
01: 1 location available
10: 2 locations available
n: n locations available (0 ≤ n ≤ 8)

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]**: Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0000: Periodic Tx FIFO is full
0001: 1 word available
0010: 2 words available
n: n words available (where 0 ≤ n ≤ PTXFD)

Others: Reserved
61.15.28 OTG host all channels interrupt register (OTG_HAINT)

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in Figure 900. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears the corresponding host channel-x interrupt register.

Address offset: 0x414
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 HAIN[15:0]: Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

61.15.29 OTG host all channels interrupt mask register (OTG_HAINTMSK)

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

Address offset: 0x418
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 HAINM[15:0]: Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15
This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in Figure 900. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

Address offset: 0x440
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>PSPD[1:0]: Port speed</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Indicates the speed of the device attached to this port.</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>01: Full speed</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>10: Low speed</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>11: Reserved</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>PTCTL[3:0]: Port test control</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>The application writes a nonzero value to this field to put</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>the port into a Test mode, and the corresponding pattern is</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>signaled on the port.</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0000: Test mode disabled</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0001: Test_J mode</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0010: Test_K mode</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0011: Test_SE0_NAK mode</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0100: Test_Packet mode</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0101: Test_Force_Enable</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Others: Reserved</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>PPWR: Port power</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>The application uses this field to control power to this</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>port, and the core clears this bit on an overcurrent</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>condition.</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>PLSTS[1:0]: Port line status</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Indicates the current logic level USB data lines</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Bit 10: Logic level of OTG_DP</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Bit 11: Logic level of OTG_DM</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 PSPD[1:0]: Port speed
- 01: Full speed
- 10: Low speed
- 11: Reserved

Bits 16:13 PTCTL[3:0]: Port test control
- 0000: Test mode disabled
- 0001: Test_J mode
- 0010: Test_K mode
- 0011: Test_SE0_NAK mode
- 0100: Test_Packet mode
- 0101: Test_Force_Enable
- Others: Reserved

Bit 12 PPWR: Port power
- 0: Power off
- 1: Power on

Bits 11:10 PLSTS[1:0]: Port line status
- Indicates the current logic level USB data lines
- Bit 10: Logic level of OTG_DP
- Bit 11: Logic level of OTG_DM

Bit 9 Reserved, must be kept at reset value.
Bit 8 **PRST**: Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

- 0: Port not in reset
- 1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

- High speed: 50 ms
- Full speed/Low speed: 10 ms

Bit 7 **PSUSP**: Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wakeup signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wakeup detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).

- 0: Port not in suspend mode
- 1: Port in suspend mode

Bit 6 **PRES**: Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wakeup sequence, as indicated by the port resume/remote wakeup detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

- 0: No resume driven
- 1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follows:

1. The application sets this bit to drive resume signaling on the port.
2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.
3. If the core detects a USB remote wakeup sequence, as indicated by the port L1Resume/Remote L1Wakeup detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume. This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG**: Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA**: Port overcurrent active

Indicates the overcurrent condition of the port.

- 0: No overcurrent condition
- 1: Overcurrent condition

Bit 3 **PENCHNG**: Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.
Bit 2 **PENA**: Port enable
A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.
0: Port disabled
1: Port enabled

Bit 1 **PCDET**: Port connect detected
The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS**: Port connect status
0: No device is attached to the port
1: A device is attached to the port

### 61.15.31 OTG host channel x characteristics register (OTG_HCCHARx)

**Address offset**: 0x500 + 0x20 * x, (x = 0 to 11)

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>EPDIR</td>
<td>EPNUM[3:0]</td>
<td>MPSIZ[10:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31: CHENA**: Channel enable
This field is set by the application and cleared by the OTG host.
0: Channel disabled
1: Channel enabled

**Bit 30: CHDIS**: Channel disable
The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

**Bit 29: ODDFRM**: Odd frame
This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.
0: Even frame
1: Odd frame

**Bits 28:22: DAD[6:0]**: Device address
This field selects the specific device serving as the data source or sink.
Bits 21:20  **MCNT[1:0]:** Multicount  
This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used  
00: Reserved. This field yields undefined results  
01: 1 transaction  
10: 2 transactions per frame to be issued for this endpoint  
11: 3 transactions per frame to be issued for this endpoint  
*Note: This field must be set to at least 01.*

Bits 19:18  **EPTYP[1:0]:** Endpoint type  
Indicates the transfer type selected.  
00: Control  
01: Isochronous  
10: Bulk  
11: Interrupt  

Bit 17  **LSDEV:** Low-speed device  
This field is set by the application to indicate that this channel is communicating to a low-speed device.  

Bit 16  Reserved, must be kept at reset value.  

Bit 15  **EPDIR:** Endpoint direction  
Indicates whether the transaction is IN or OUT.  
0: OUT  
1: IN  

Bits 14:11  **EPNUM[3:0]:** Endpoint number  
Indicates the endpoint number on the device serving as the data source or sink.  

Bits 10:0  **MPSIZ[10:0]:** Maximum packet size  
Indicates the maximum packet size of the associated endpoint.

### 61.15.32  **OTG host channel x interrupt register (OTG_HCINTx)**

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 900*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.  

Address offset: 0x508 + 0x20 * x, (x = 0 to 11)  
Reset value: 0x0000 0000
Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR**: Data toggle error.

Bit 9 **FRMOR**: Frame overrun.

Bit 8 **BBERR**: Babble error.

Bit 7 **TXERR**: Transaction error.
   Indicates one of the following errors occurred on the USB.
   CRC check failure
   Timeout
   Bit stuff error
   False EOP

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACK**: ACK response received/transmitted interrupt.

Bit 4 **NAK**: NAK response received interrupt.

Bit 3 **STALL**: STALL response received interrupt.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHH**: Channel halted.
   Indicates the transfer completed abnormally either because of any USB transaction error or
   in response to disable request by the application.

Bit 0 **XFRC**: Transfer completed.
   Transfer completed normally without any errors.

### 61.15.33 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

This register reflects the mask for each channel status described in the previous section.

Address offset: 0x50C + 0x20 * x, (x = 0 to 11)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-------</td>
</tr>
</tbody>
</table>
| rw  | rw  | rw  | rw  | rw  | 0: Masked interrupt
|     |     |     |     |     | 1: Unmasked interrupt

Bit 10 **DTERRM**: Data toggle error mask.

0: Masked interrupt
1: Unmasked interrupt

Bit 9 **FRMORM**: Frame overrun mask.

0: Masked interrupt
1: Unmasked interrupt
Bit 8 **BBERM**: Babble error mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 7 **TXERM**: Transaction error mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 6 Reserved, must be kept at reset value.

Bit 5 **ACKM**: ACK response received/transmitted interrupt mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 4 **NAKM**: NAK response received interrupt mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 3 **STALLM**: STALL response received interrupt mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CHHM**: Channel halted mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed mask
   0: Masked interrupt
   1: Unmasked interrupt

### 61.15.34 OTG host channel x transfer size register (OTG_HCTSIZx)

**Address offset**: 0x510 + 0x20 * x, (x = 0 to 11)

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XFRSIZ[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31 DOPNG: Do Ping
This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.
Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 DPID[1:0]: Data PID
The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.
00: DATA0
10: DATA1
11: SETUP (control) / reserved (non-control)

Bits 28:19 PKTCNT[9:0]: Packet count
This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).
The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 XFRSIZ[18:0]: Transfer size
For an OUT, this field is the number of data bytes the host sends during the transfer.
For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

61.15.35 Device-mode registers
These registers must be programmed every time the core changes to device mode

61.15.36 OTG device configuration register (OTG_DCFG)
This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.
Address offset: 0x800
Reset value: 0x0220 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 ERRATIM: Erratic error interrupt mask
1: Mask early suspend interrupt on erratic error
0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.
Bits 12:11 **PFIVL[1:0]**: Periodic frame interval
Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.
- 00: 80% of the frame interval
- 01: 85% of the frame interval
- 10: 90% of the frame interval
- 11: 95% of the frame interval

Bits 10:4 **DAD[6:0]**: Device address
The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHKS**: Non-zero-length status OUT handshake
The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer’s status stage.
- 1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
- 0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]**: Device speed
Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.
- 00: Reserved
- 01: Reserved
- 10: Reserved
- 11: Full speed (USB 1.1 transceiver clock is 48 MHz)

### 61.15.37 OTG device control register (OTG_DCTL)

**Address offset:** 0x804
**Reset value:** 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:19  Reserved, must be kept at reset value.

Bit 18  **DBESLRJCT**: Deep sleep BESL reject
Core rejects LPM request with BESL value greater than BESL threshold programmed.
NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12  Reserved, must be kept at reset value.

Bit 11  **POPRGDNE**: Power-on programming done
The application uses this bit to indicate that register programming is completed after a wakeup from power down mode.

Bit 10  **CGONAK**: Clear global OUT NAK
Writing 1 to this field clears the Global OUT NAK.

Bit 9  **SGONAK**: Set global OUT NAK
Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8  **CGINAK**: Clear global IN NAK
Writing 1 to this field clears the Global IN NAK.

Bit 7  **SGINAK**: Set global IN NAK
Writing 1 to this field sets the Global non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4  **TCTL[2:0]**: Test control
000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3  **GONSTS**: Global OUT NAK status
0: A handshake is sent based on the FIFO status and the NAK and STALL bit settings.
1: No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.
Bit 2 **GINSTS**: Global IN NAK status

0: A handshake is sent out based on the data availability in the transmit FIFO.

1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS**: Soft disconnect

The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.

0: Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.

1: The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG**: Remote wakeup signaling

When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.

If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs (TL1DevDrvResume) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

*Table 661* contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

<table>
<thead>
<tr>
<th>Operating speed</th>
<th>Device state</th>
<th>Minimum duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full speed</td>
<td>Suspended</td>
<td>1 ms + 2.5 µs</td>
</tr>
<tr>
<td>Full speed</td>
<td>Idle</td>
<td>2.5 µs</td>
</tr>
<tr>
<td>Full speed</td>
<td>Not Idle or suspended (Performing transactions)</td>
<td>2.5 µs</td>
</tr>
</tbody>
</table>
61.15.38 OTG device status register (OTG_DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

Address offset: 0x808
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>EERR</td>
<td>ENUMSPD[1:0]</td>
<td>SUSPSTS</td>
</tr>
</tbody>
</table>

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 DEVLNSTS[1:0]: Device line status
Indicates the current logic level USB data lines.
- Bit [23]: Logic level of D+
- Bit [22]: Logic level of D-

Bits 21:8 FNSOF[13:0]: Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 EERR: Erratic error
The core sets this bit to report any erratic errors.
Due to erratic errors, the OTG_FS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 ENUMSPD[1:0]: Enumerated speed
Indicates the speed at which the OTG_FS controller has come up after speed detection through a chirp sequence.
- 11: Full speed using embedded FS PHY
- Others: reserved

Bit 0 SUSPSTS: Suspend status
In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:
- When there is an activity on the USB data lines
- When the application writes to the remote wakeup signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).
61.15.39  OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x810

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 13</td>
<td><strong>NAKM</strong>: NAK interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 12:7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 6</td>
<td><strong>INEPNEM</strong>: IN endpoint NAK effective mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 5</td>
<td><strong>INEPNMM</strong>: IN token received with EP mismatch mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 4</td>
<td><strong>ITTXFEMSK</strong>: IN token received when Tx FIFO empty mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>TOM</strong>: Timeout condition mask (Non-isochronous endpoints)</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 1</td>
<td><strong>EPDM</strong>: Endpoint disabled interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 0</td>
<td><strong>XFRCM</strong>: Transfer completed interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
</tbody>
</table>
61.15.40 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x814
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 13</td>
<td>NAKMSK: NAK interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 12</td>
<td>BERRM: Babble error interrupt mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bits 11:10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 8</td>
<td>OUTPKTERRM: Out packet error mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 6</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 5</td>
<td>STSPHSRXM: Status phase received for control write mask</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 4</td>
<td>OTEPDM: OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 3</td>
<td>STUPM: STUPM: SETUP phase done mask. Applies to control endpoints only.</td>
</tr>
<tr>
<td></td>
<td>0: Masked interrupt</td>
</tr>
<tr>
<td></td>
<td>1: Unmasked interrupt</td>
</tr>
</tbody>
</table>
Bit 2  Reserved, must be kept at reset value.

Bit 1  **EPDM**: Endpoint disabled interrupt mask
       0: Masked interrupt
       1: Unmasked interrupt

Bit 0  **XFRCM**: Transfer completed interrupt mask
       0: Masked interrupt
       1: Unmasked interrupt

### 61.15.41 OTG device all endpoints interrupt register (OTG_DAINT)

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

Address offset: 0x818

Reset value: 0x0000 0000

<p>| | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>OEPINT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>IEPINT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Bits 31:16 **OEPINT[15:0]**: OUT endpoint interrupt bits

One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]**: IN endpoint interrupt bits

One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.
61.15.42 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

Address offset: 0x81C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEPM[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
| Bits 31:16 | **OEPM[15:0]**: OUT EP interrupt mask bits  
One per OUT endpoint:  
Bit 16 for OUT EP 0, bit 19 for OUT EP 3  
0: Masked interrupt  
1: Unmasked interrupt |
| Bits 15:0 | **IEPM[15:0]**: IN EP interrupt mask bits  
One bit per IN endpoint:  
Bit 0 for IN EP 0, bit 3 for IN EP 3  
0: Masked interrupt  
1: Unmasked interrupt |

61.15.43 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

Address offset: 0x834
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>INEPTXFEM[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 \texttt{INEPTXFEM[15:0]}: IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG\_DEPINTx.

TXFE interrupt one bit per IN endpoint:
- Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
- 0: Masked interrupt
- 1: Unmasked interrupt

### 61.15.44 OTG device each endpoint interrupt register (OTG\_DEACHINT)
Address offset: 0x0838
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 17 \texttt{OEP1INT}: OUT endpoint 1 interrupt bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 \texttt{IEP1INT}: IN endpoint 1 interrupt bit

Bit 0 Reserved, must be kept at reset value.

### 61.15.45 OTG device each endpoint interrupt mask register (OTG\_DEACHINTMSK)
There is one interrupt bit for endpoint 1 IN and one interrupt bit for endpoint 1 OUT.
Address offset: 0x083C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 17 \texttt{OEP1INT}: OUT endpoint 1 interrupt bit

Bits 16:2 Reserved, must be kept at reset value.

Bit 1 \texttt{IEP1INT}: IN endpoint 1 interrupt bit

Bit 0 Reserved, must be kept at reset value.
**61.15.46 OTG device each IN endpoint-1 interrupt mask register (OTG_HS_DIEPEACHMSK1)**

This register works with the OTG_DIEPINT1 register to generate a dedicated interrupt OTG_HS_EP1_IN for endpoint #1. The IN endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x844

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-14</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 13</td>
<td><strong>NAKM</strong>: NAK interrupt mask</td>
</tr>
<tr>
<td>0</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1</td>
<td>Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 12-10</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 9</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 8</td>
<td><strong>TXFURM</strong>: FIFO underrun mask</td>
</tr>
<tr>
<td>0</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1</td>
<td>Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 6</td>
<td><strong>INEPNEM</strong>: IN endpoint NAK effective mask</td>
</tr>
<tr>
<td>0</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1</td>
<td>Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4</td>
<td><strong>ITTXFEMSK</strong>: IN token received when Tx FIFO empty mask</td>
</tr>
<tr>
<td>0</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1</td>
<td>Unmasked interrupt</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>TOM</strong>: Timeout condition mask (Non-isochronous endpoints)</td>
</tr>
<tr>
<td>0</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1</td>
<td>Unmasked interrupt</td>
</tr>
</tbody>
</table>
Bit 2 **AHBERRM**: AHB error mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 1 **EPDM**: Endpoint disabled interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

### 61.15.47 OTG device each OUT endpoint-1 interrupt mask register (OTG_HS_DOEPEACHMSK1)

This register works with the OTG_DOEPINT1 register to generate a dedicated interrupt OTG_HS_EP1_OUT for endpoint #1. The OUT endpoint interrupt for a specific status in the OTG_DOEPINT1 register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x884
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>NYETMSK: NYET interrupt mask</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>30</td>
<td>NAKMSK: NAK interrupt mask</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>29</td>
<td>BERRM: Babble error interrupt mask</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK**: NYET interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 13 **NAKMSK**: NAK interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM**: Out packet error mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.
Bit 6 **B2BSTUPM:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** STUPM: SETUP phase done mask
Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

### 61.15.48 OTG device control IN endpoint 0 control register (OTG_DIEPCTL0)

This section describes the OTG_DIEPCTL0 register for USB_OTG FS. Nonzero control endpoints use registers for endpoints 1–3.

Address offset: 0x900
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

---
Bit 31 EPENA: Endpoint enable
   The application sets this bit to start transmitting data on the endpoint 0.
   The core clears this bit before setting any of the following interrupts on this endpoint:
   - Endpoint disabled
   - Transfer completed

Bit 30 EPDIS: Endpoint disable
   The application sets this bit to stop transmitting data on an endpoint, even before the
   transfer for that endpoint is complete. The application must wait for the endpoint disabled
   interrupt before treating the endpoint as disabled. The core clears this bit before setting the
   endpoint disabled interrupt. The application must set this bit only if endpoint enable is
   already set for this endpoint.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 SNAK: Set NAK
   A write to this bit sets the NAK bit for the endpoint.
   Using this bit, the application can control the transmission of NAK handshakes on an
   endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on
   that endpoint.

Bit 26 CNAK: Clear NAK
   A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 TXFNUM[3:0]: Tx FIFO number
   This value is set to the FIFO number that is assigned to IN endpoint 0.

Bit 21 STALL: STALL handshake
   The application can only set this bit, and the core clears it when a SETUP token is received
   for this endpoint. If a NAK bit, a Global IN NAK or Global OUT NAK is set along with this bit,
   the STALL bit takes priority.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 EPTYP[1:0]: Endpoint type
   Hardcoded to ‘00’ for control.

Bit 17 NAKSTS: NAK status
   Indicates the following:
   0: The core is transmitting non-NAK handshakes based on the FIFO status
   1: The core is transmitting NAK handshakes on this endpoint.
   When this bit is set, either by the application or core, the core stops transmitting data, even
   if there are data available in the Tx FIFO. Irrespective of this bit’s setting, the core always
   responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.
Bit 15 **USBAEP**: USB active endpoint
This bit is always set to 1, indicating that control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]**: Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint.
- 00: 64 bytes
- 01: 32 bytes
- 10: 16 bytes
- 11: 8 bytes

### 61.15.49 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)
Valid for INT/BULK endpoints, see next section for ISO endpoints.
The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 1 to 5)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>EPENA</strong>: Endpoint enable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The application sets this bit to start transmitting data on an endpoint.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>The core clears this bit before setting any of the following interrupts on this endpoint:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>– SETUP phase done</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Endpoint disabled</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>– Transfer completed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td><strong>EPDIS</strong>: Endpoint disable</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td><strong>SD1PID</strong>: Set DATA1 PID</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td><strong>SD0PID</strong>: Set DATA0 PID</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Applies to interrupt/bulk IN endpoints only. Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bit 27 **SNAK**: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]**: Tx FIFO number
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL**: STALL handshake
Applies to non-control, non-isochronous IN endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.
Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]**: Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS**: NAK status
It indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.
For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID**: Endpoint data PID
Applies to interrupt/bulk IN endpoints only.
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.
0: DATA0
1: DATA1
Bit 15 **USBAEP**: USB active endpoint
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11
Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

61.15.50 **OTG device IN endpoint x control register [alternate]**
(OTG_DIEPCTLx)
Valid for ISO endpoints, see previous section for INT/BULK endpoints.
The application uses this register to control the behavior of each logical endpoint other than endpoint 0.
Address offset: 0x900 + 0x20 * x, (x = 1 to 5)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPENA</td>
<td>EPDIS</td>
<td>SODDFRM</td>
<td>SEVNFRM</td>
<td>SNAK</td>
<td>CNAK</td>
<td>TXNUM[3:0]</td>
<td>STALL</td>
<td>Res.</td>
<td>EPTYP[1:0]</td>
<td>NAKSTS</td>
<td>EO</td>
<td>NUM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPENA</td>
<td>EPDIS</td>
<td>SODDFRM</td>
<td>SEVNFRM</td>
<td>SNAK</td>
<td>CNAK</td>
<td>TXNUM[3:0]</td>
<td>STALL</td>
<td>Res.</td>
<td>EPTYP[1:0]</td>
<td>NAKSTS</td>
<td>EO</td>
<td>NUM</td>
</tr>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **EPENA**: Endpoint enable
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS**: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM**: Set odd frame
Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM**: Set even frame
Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.
Bit 27 **SNAK:** Set NAK  
A write to this bit sets the NAK bit for the endpoint.  
Using this bit, the application can control the transmission of NAK handshakes on an  
endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt,  
or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK  
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number  
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint  
must be programmed to a separate FIFO number.  
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake  
Applies to non-control, non-isochronous IN endpoints only (access type is rw).  
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK  
bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.  
Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type  
This is the transfer type supported by this logical endpoint.  
00: Control  
01: Isochronous  
10: Bulk  
11: Interrupt

Bit 17 **NAKSTS:** NAK status  
It indicates the following:  
0: The core is transmitting non-NAK handshakes based on the FIFO status.  
1: The core is transmitting NAK handshakes on this endpoint.  
When either the application or the core sets this bit:
For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint,  
even if there are data available in the Tx FIFO.  
For isochronous IN endpoints: The core sends out a zero-length data packet, even if there  
are data available in the Tx FIFO.  
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an  
ACK handshake.

Bit 16 **EONUM:** Even/odd frame  
Applies to isochronous IN endpoints only.  
Indicates the frame number in which the core transmits/receives isochronous data for this  
endpoint. The application must program the even/odd frame number in which it intends to  
transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM  
fields in this register.  
0: Even frame  
1: Odd frame
This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in Figure 900. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0x908 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0080

| Bit 31:14 | Reserved, must be kept at reset value. |
| Bit 13 | NAK: NAK input |
| The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO. |
| Bit 12 | Reserved, must be kept at reset value. |
| Bit 11 | PKTDRPSTS: Packet dropped status |
| This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt. |
| Bits 10:8 | Reserved, must be kept at reset value. |
| Bit 7 | TXFE: Transmit FIFO empty |
| This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG). |
Bit 6 **INEPNE**: IN endpoint NAK effective  
This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.
This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.
This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch  
Indicates that the data in the top of the non-periodic Tx FIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE**: IN token received when Tx FIFO is empty  
Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC**: Timeout condition  
Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD**: Endpoint disabled interrupt  
This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC**: Transfer completed interrupt  
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

### 61.15.52 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

Address offset: 0x910  
Reset value: 0x0000 0000
Bits 31:21  Reserved, must be kept at reset value.

Bits 20:19  **PKTCNT[1:0]:** Packet count  
Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.  
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7  Reserved, must be kept at reset value.

Bits 6:0  **XFRSIZ[6:0]:** Transfer size  
Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.  
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

### 61.15.53  OTG device IN endpoint transmit FIFO status register  
(OTG_DTXFSTSx)

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

Address offset: 0x918 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0200

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available  
Indicates the amount of free space available in the endpoint Tx FIFO.  
Values are in terms of 32-bit words:  
0: Endpoint Tx FIFO is full  
1: 1 word available  
2: 2 words available  
n: n words available  
Others: Reserved
61.15.54 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0x910 + 0x20 * x, (x = 1 to 5)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30-29</th>
<th>Bit 28-19</th>
<th>Bit 18-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>14</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>13</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>12</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]: Multi count**
For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.
- 01: 1 packet
- 10: 2 packets
- 11: 3 packets

Bits 28:19 **PKTCNT[9:0]: Packet count**
Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]: Transfer size**
This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.
61.15.55  **OTG device control OUT endpoint 0 control register**  
(OTG_DOEPCTL0)

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–3.

Address offset: 0xB00

Reset value: 0x0000 8000

| Bit 31 | EPENA: Endpoint enable  
| The application sets this bit to start transmitting data on endpoint 0.  
| The core clears this bit before setting any of the following interrupts on this endpoint:  
| – SETUP phase done  
| – Endpoint disabled  
| – Transfer completed  

| Bit 30 | EPDIS: Endpoint disable  
| The application cannot disable control OUT endpoint 0.  

| Bits 29:28 | Reserved, must be kept at reset value.  

| Bit 27 | SNAK: Set NAK  
| A write to this bit sets the NAK bit for the endpoint.  
| Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.  

| Bit 26 | CNAK: Clear NAK  
| A write to this bit clears the NAK bit for the endpoint.  

| Bits 25:22 | Reserved, must be kept at reset value.  

| Bit 21 | STALL:STALL handshake  
| The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.  

| Bit 20 | SNPM: Snoop mode  
| This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.  

| Bits 19:18 | EPTYP[1:0]: Endpoint type  
| Hardcoded to 2'b00 for control.  

---

## USB on-the-go full-speed (OTG_FS)  
RM0477
Bit 17 **NAKSTS**: NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP**: USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]**: Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.

- 00: 64 bytes
- 01: 32 bytes
- 10: 16 bytes
- 11: 8 bytes

61.15.56 **OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)**

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in Figure 900. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0xB08 + 0x20 * x, (x = 0 to 5)

Reset value: 0x0000 0080

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK**: NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR**: Babble error interrupt

The core generates this interrupt when babble is received for the endpoint.
Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **STSPHSRX**: Status phase received for control write

This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG_FS has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS**: OUT token received when endpoint disabled

 Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP**: SETUP phase done

 Applies to control OUT endpoint only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **EPDISD**: Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC**: Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

### 61.15.57 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCCTL0 registers (EPENA bit in OTG_DOEPCCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–5.

Address offset: 0xB10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

2990/3791
Bit 31  Reserved, must be kept at reset value.

Bits 30:29  STUPCNT[1:0]: SETUP packet count
   This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
   01: 1 packet
   10: 2 packets
   11: 3 packets

Bits 28:20  Reserved, must be kept at reset value.

Bit 19  PKTCNT: Packet count
   This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7  Reserved, must be kept at reset value.

Bits 6:0  XFRSIZ[6:0]: Transfer size
   Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
   The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

### 61.15.58  OTG device OUT endpoint x control register [alternate]
(OTG_DOEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 5)

Reset value: 0x0000 0000
Bit 31  **EPENA**: Endpoint enable
Applies to IN and OUT endpoints.
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30  **EPDIS**: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before
the transfer for that endpoint is complete. The application must wait for the endpoint
disabled interrupt before treating the endpoint as disabled. The core clears this bit before
setting the endpoint disabled interrupt. The application must set this bit only if endpoint
enable is already set for this endpoint.

Bit 29  **SD1PID**: Set DATA1 PID
Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the endpoint
data PID (DPID) field in this register to DATA1.

Bit 28  **SD0PID**: Set DATA0 PID
Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27  **SNAK**: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an
endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt,
or after a SETUP is received on the endpoint.

Bit 26  **CNAK**: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22  Reserved, must be kept at reset value.

Bit 21  **STALL**: STALL handshake
Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK
bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes
priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received
for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit,
the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to
SETUP data packets with an ACK handshake.

Bit 20  **SNPM**: Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check
the correctness of OUT packets before transferring them to application memory.

Bits 19:18  **EPTYP[1:0]**: Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt
Bit 17 **NAKSTS**: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID**: Endpoint data PID
Applies to interrupt/bulk OUT endpoints only.
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID register field to program either DATA0 or DATA1 PID.
0: DATA0
1: DATA1

Bit 15 **USBAEP**: USB active endpoint
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

61.15.59 **OTG device OUT endpoint x control register [alternate]**
(OTG_DOEPCTLx)
Valid for ISO endpoints, see previous section for INT/BULK endpoints.
The application uses this register to control the behavior of each logical endpoint other than endpoint 0.
Address offset: 0xB00 + 0x20 * x, (x = 1 to 5)
Reset value: 0x0000 0000

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


Bit 31 **EPENA**: Endpoint enable

Applies to IN and OUT endpoints.
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 **EPDIS**: Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before
the transfer for that endpoint is complete. The application must wait for the endpoint
disabled interrupt before treating the endpoint as disabled. The core clears this bit before
setting the endpoint disabled interrupt. The application must set this bit only if endpoint
enable is already set for this endpoint.

Bit 29 **SODDFRM**: Set odd frame

Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd
frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM**: Set even frame

Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK**: Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an
endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt,
or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL**: STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK
bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes
priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received
for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit,
the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to
SETUP data packets with an ACK handshake.

Bit 20 **SNPM**: Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check
the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]**: Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt
Bit 17 NAKSTS: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 EONUM: Even/odd frame
Applies to isochronous IN and OUT endpoints only.
Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.
0: Even frame
1: Odd frame

Bit 15 USBAEP: USB active endpoint
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 MPSIZ[10:0]: Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

61.15.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)
The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.
Address offset: 0xB10 + 0x20 * x, (x = 1 to 5)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:24</th>
<th>Bits 23:16</th>
<th>Bits 15:8</th>
<th>Bits 7:4</th>
<th>Bits 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

XFRSIZ[15:0]
Bit 31  Reserved, must be kept at reset value.

Bits 30:29  RXDPID[1:0]:
  Condition: isochronous OUT endpoints
  Received data PID
  This is the data PID received in the last packet for this endpoint.
  00: DATA0
  01: DATA2
  10: DATA1
  11: MDATA
  Condition: control OUT endpoints
  STUPCNT[1:0]: SETUP packet count
  This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
  01: 1 packet
  10: 2 packets
  11: 3 packets

Bits 28:19  PKTCNT[9:0]: Packet count
  Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
  This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0  XFRSIZ[18:0]: Transfer size
  This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
  The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

61.15.61  OTG power and clock gating control register (OTG_PCGCCTL)

This register is available in host and device modes.

Address offset: 0xE00

Reset value: 0x200B 8000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GTG</td>
<td>PHYS</td>
<td>SUSP</td>
<td>Res</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Res</td>
<td>HCLK</td>
<td>CLK</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 7  SUSP: Deep Sleep
  This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6  PHYSLEEP: PHY in Sleep
  This bit indicates that the PHY is in the Sleep state.
Bit 5 **EN1GTG**: Enable sleep clock gating
   When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP**: PHY suspended
   Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>EN1GTG</strong></td>
<td><strong>PHYSUSP</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 1 **GATEHCLK**: Gate HCLK
   The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK**: Stop PHY clock
   The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

### 61.15.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1)

This register is available in host and device modes.

Address offset: 0xE04

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Access</th>
<th>Access</th>
<th>Access</th>
<th>Access</th>
<th>Access</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>Description</th>
<th>Access</th>
<th>Access</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>RAMGATEEN</strong>: Enable RAM clock gating</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 2:1 **CNTGATECLK[1:0]**: Counter for clock gating
   Indicates to the controller how many PHY clock cycles and AHB clock cycles of 'IDLE' (no activity) the controller waits for before gating the respective PHY and AHB clocks internal to the controller.
   00: 64 clocks
   01: 128 clocks
   10: Reserved
   11: Reserved

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>GATEEN</strong>: Enable active clock gating</td>
<td></td>
</tr>
</tbody>
</table>

   The application programs GATEEN to enable active clock gating feature for the PHY and AHB clocks.
## 61.15.63 OTG_FS register map

The table below gives the USB OTG register map and reset values.

| Offset | Register name          | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | OTG_GOTGCTL            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 0   | 0   | 0   | 0   | 0   | 1   | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x004  | OTG_GOTGINT            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x008  | OTG_GAHBCFG            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x00C  | OTG_GUSBCFG            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x010  | OTG_GRSTCTL            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 1   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x014  | OTG_GINTSTS            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x018  | OTG_GINTMSK            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value            | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |

**Table 662. OTG_FS register map and reset values**

**Offset**: Address offset from the base address of the USB OTG controller.

**Reset value**: The reset value of the register.

**Register name**: The name of the register.

**Register map**: The map of the register, showing the control bits and their reset values.
### Table 662. OTG_FS register map and reset values (continued)

| Offset  | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x01C   | OTG_GRXSTSR (Device mode) |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x020   | OTG_GRXSTSP (Device mode) |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x024   | OTG_GRXFSIZ            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x028   | OTG_HNPTXFSIZ/OTG_DIEPTXF0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x02C   | OTG_HNPTXSTS           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x038   | OTG_GCCFG              |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x03C   | OTG_CID                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x054   | OTG_GLPMCFG            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x100   | OTG_HPTXFSIZ           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x104   | OTG_DIEPTXF1           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value            | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |

Reset value: 0 0 0 0 0 X X X X
### Table 662. OTG_FS register map and reset values (continued)

| Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset | Register name | Offset |
|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|---------------|--------|
| 0x114  | OTG_DIEPTXF5  | INEPTXFD | INEPTXSA      | 0x400  | OTG_HCFCFG    | FSL    | FSLS          | 0x404  | OTG_HFIR      | FD     | FRIVL         | 0x408  | OTG_HFNUM     | FTREM  | FNRM          | 0x410  | OTG_HFPTXSTS  | PTXQTOP| PTXTQSAV      | PTXFSAVL| 0x414  | OTG_HAINT     | HAIN   | HAINTM        | 0x418  | OTG_HAINTMSK  | HAINTM | HAINTM        | 0x440  | OTG_HPRT      | PSP     | PTCL          | 0x500  | OTG_HCCHAR0   | CHRA   | CHRC          | 0x508  | OTG_HCINT0    | DTErr   | DTErr         | 0x50B  | OTG_HCINT0    | DTErr   | DTErr         |
|        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |
|        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |        |               |

- **Reset value**: 00000000000000000000000000000000
- **FSL**: FS LS
- **PCS**: PCS

**Note**: The table continues with similar entries for each register, showing the offset, register name, and reset values.
Table 662. OTG_FS register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x510</td>
<td>OTG_HCCTSIZ0</td>
<td>DOPNG</td>
<td>DPID</td>
<td>PKTCNT</td>
<td>XFRSIZ</td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Iterating preceding block of registers starting at offset 0x500.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x560</td>
<td>OTG_HCCHAR11</td>
<td>CCHAR</td>
<td>GCHAR</td>
<td>OGOPRM</td>
<td>DAD</td>
<td>MCNT</td>
<td>ETYPE</td>
<td>LSBEV</td>
<td>EPDIR</td>
<td>EPNUM</td>
<td>MPSIZ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x660</td>
<td>OTG_HCINT11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x668</td>
<td>OTG_HCINTMSK11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x700</td>
<td>OTG_HCCTSIZ11</td>
<td>DOPNG</td>
<td>DPID</td>
<td>PKTCNT</td>
<td>XFRSIZ</td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x800</td>
<td>OTG_DCFG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x804</td>
<td>OTG_DCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x808</td>
<td>OTG_DSTS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x810</td>
<td>OTG_DIEPMSK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 662. OTG_FS register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| 0x814  | OTG_DOEPMSK   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x818  | OTG DAINT     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x81C  | OTG_DAITMSK   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x834  | OTG_DIEPEPMASK|    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x838  | OTG_DIECHINT  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x83C  | OTG_DIECHINT  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x844  | OTG_HS_DIEPEACH_MSK1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x884  | OTG_HS_DIEPEACH_MSK1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x900  | OTG_DIEPCTL0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x908  | OTG_DIEPINT0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x910  | OTG_DIEPIFSIZ0 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
Table 662. OTG_FS register map and reset values (continued)

| Offset | Register name      | Offset Register Name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|--------------------|----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x910   | OTG_DTXFSTS0       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        | INEPTFSAV            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x920   | OTG_DIEPCTL1       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | (INT/BULK)         |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x928   | OTG_DIEPINT1       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x930   | OTG_DIEPTSIZ1      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x938   | OTG_DTXFSTS1       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Iterating preceding block of registers starting at offset 0x920. |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9A0   | OTG_DIEPCTL5       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | (INT/BULK)         |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9A8   | OTG_DIEPINT5       |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x9B0   | OTG_DIEPTSIZ5      |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value        |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         |                    |                      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 662. OTG_FS register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>EPENA</th>
<th>EPDIS</th>
<th>SD1PID</th>
<th>SD0PID</th>
<th>SNAK</th>
<th>CNAK</th>
<th>STALL</th>
<th>SNPM</th>
<th>EPTYP</th>
<th>NAKSTS</th>
<th>USBAP</th>
<th>DIPL</th>
<th>EONUM</th>
<th>STSPHSRX</th>
<th>OTEPDIS</th>
<th>STUP</th>
<th>EPDISD</th>
<th>XFRC</th>
<th>MPSIZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB00</td>
<td>OTG_DOEPCTL0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>1 0 0</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td>0xB08</td>
<td>OTG_DOEPINT0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td>0xB10</td>
<td>OTG_DOEPSTSIZ0</td>
<td>STU</td>
<td>CNT</td>
<td>PKTCNT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td>0xB20</td>
<td>OTG_DOEPCTL1</td>
<td>EPENA</td>
<td>EPDIS</td>
<td>SD1PID</td>
<td>SD0PID</td>
<td>SNAK</td>
<td>CNAK</td>
<td>STALL</td>
<td>SNPM</td>
<td>EPTYP</td>
<td>NAKSTS</td>
<td>USBAP</td>
<td>DIPL</td>
<td>EONUM</td>
<td>STSPHSRX</td>
<td>OTEPDIS</td>
<td>STUP</td>
<td>EPDISD</td>
<td>XFRC</td>
<td>MPSIZ</td>
</tr>
<tr>
<td></td>
<td>(INT/BULK)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td>0xB28</td>
<td>OTG_DOEPINT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td>0xB30</td>
<td>OTG_DOEPSTSIZ1</td>
<td>PKTCNT</td>
<td></td>
<td>XFRSIZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

- Iterating preceding block of registers starting at offset 0xB20.

0xBA0  | OTG_DOEPCTL5       | EPENA | EPDIS | SD1PID | SD0PID | SNAK | CNAK | STALL | SNPM | EPTYP | NAKSTS | USBAP | DIPL | EONUM | STSPHSRX | OTEPDIS | STUP | EPDISD | XFRC | MPSIZ |
|        | (INT/BULK)          |       |       |        |        |      |      |       |      |      |        |       |      |      |            |         |      |        |      |       |
|        | Reset value         | 0 0 0 | 0 0 0 | 0 0 0 0| 0 0 0 0| 0 0 0| 0 0 0|      |      |      |        |       |      |      |            |         |      |        |      | 0 0 0 |
| 0xBA8  | OTG_DOEPCTL5       | EPENA | EPDIS | SD1PID | SD0PID | SNAK | CNAK | STALL | SNPM | EPTYP | NAKSTS | USBAP | DIPL | EONUM | STSPHSRX | OTEPDIS | STUP | EPDISD | XFRC | MPSIZ |
|        | (ISO)               |       |       |        |        |      |      |       |      |      |        |       |      |      |            |         |      |        |      |       |
|        | Reset value         | 0 0 0 | 0 0 0 | 0 0 0 0| 0 0 0 0| 0 0 0| 0 0 0|      |      |      |        |       |      |      |            |         |      |        |      | 0 0 0 |
Refer to **Section 2.3 on page 149** for the register boundary addresses.
61.16 **OTG_FS programming model**

61.16.1 **Core initialization**

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG_FS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_FS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level

2. Program the following fields in the OTG_GUSBCFG register:
   - OTG_FS timeout calibration field
   - USB turnaround time field

3. The software must unmask the following bits in the OTG_GINTMSK register:
   - OTG interrupt mask
   - Mode mismatch interrupt mask

4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG_FS controller is operating in host or device mode.
61.16.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. This drives $V_{BUS}$ on the USB.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

61.16.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval
2. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
3. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
4. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
5. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS
register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion on page 3030*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

### 61.16.4 Host programming model

#### Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

#### Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_FS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG_FS host does not interrupt the transaction that has already been started on the USB.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:
1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).
3. When the application aborts a transfer before normal completion.

Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**
  The OTG_FS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG_FS host determines the actual packet size based on the programmed maximum packet size and transfer size.

![Figure 901. Transmit FIFO write task](image)

- **Reading the receive FIFO**
  The application must ignore all packet statuses other than IN data packet (bx0010).
- **Bulk and control OUT/SETUP transactions**
  
  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in Figure 903. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:
  
  - The application is attempting to send two maximum-packet-size packets (transfer size = 1,024 bytes).
  - The non-periodic transmit FIFO can hold two packets (128 bytes for FS).
  - The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**
  
  The sequence of operations in (channel 1) is as follows:
  
  1. Initialize channel 1
  2. Write the first packet for channel 1
  3. Along with the last word write, the core writes an entry to the non-periodic request queue
  4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
  5. Write the second (last) packet for channel 1
  6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
  7. In response to the XFRC interrupt, de-allocate the channel for other transfers
  8. Handling non-ACK responses
**Figure 903. Normal bulk/control OUT/SETUP**

1. The grayed elements are not relevant in the context of this figure.
The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**
  - a) Bulk/control OUT/SETUP

Unmask (NAK/TXERR/STALL/XFRC)

```c
if (XFRC)
{
  Reset Error Count
  Mask ACK
  De-allocate Channel
}
else if (STALL)
{
  Transfer Done = 1
  Unmask CHH
  Disable Channel
}
else if (NAK or TXERR )
{
  Rewind Buffer Pointers
  Unmask CHH
  Disable Channel
  if (TXERR)
  {
    Increment Error Count
    Unmask ACK
  }
  else
  {
    Reset Error Count
  }
}
else if (CHH)
{
  Mask CHH
  if (Transfer Done or (Error_count == 3))
  {
    De-allocate Channel
  }
  else
  {
    Re-initialize Channel
  }
}
```
else if (ACK)
{
  Reset Error Count
  Mask ACK
}

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

Unmask (TXERR/XFRC/BBERR/STALL/DTE_RR)

if (XFRC)
{
  Reset Error Count
  Unmask CHH
  Disable Channel
  Reset Error Count
  Mask ACK
}

else if (TXERR or BBERR or STALL)
{
  Unmask CHH
  Disable Channel
  if (TXERR)
    {
      Increment Error Count
      Unmask ACK
    }
}

else if (CHH)
{
  Mask CHH
  if (Transfer Done or (Error_count == 3))
    {
      De-allocate Channel
    }
  else
    {
      Re-initialize Channel
    }
}

else if (ACK)
{
  Reset Error Count
  Mask ACK
}
else if (DTERR)
{
    Reset Error Count
}

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 904*. See channel 2 (ch_2). The assumptions are:
  - The application is attempting to receive two maximum-packet-size packets (transfer size = 1024 bytes).
  - The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (72 bytes for FS).
  - The non-periodic request queue depth = 4.
Figure 904. Bulk/control IN transactions

1. The grayed elements are not relevant in the context of this figure.
The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. Setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in
OTG_HCCHAR1 to control. During the setup stage, the application is expected to set
the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**
  A typical interrupt OUT operation is shown in Figure 905. The assumptions are:
  - The application is attempting to send one packet in every frame (up to 1 maximum
    packet size), starting with the odd frame (transfer size = 1 024 bytes)
  - The periodic transmit FIFO can hold one packet (1 Kbyte)
  - Periodic request queue depth = 4

  The sequence of operations is as follows:
  1. Initialize and enable channel 1. The application must set the ODDFRM bit in
     OTG_HCCHAR1.
  2. Write the first packet for channel 1.
  3. Along with the last word write of each packet, the OTG_FS host writes an entry to the
     periodic request queue.
  4. The OTG_FS host attempts to send an OUT token in the next (odd) frame.
  5. The OTG_FS host generates an XFRC interrupt as soon as the last packet is
     transmitted successfully.
  6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
**Figure 905. Normal interrupt OUT**

1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**
  
  a) Interrupt OUT

  **Unmask** (NAK/TXERR/STALL/XFRC/FRMOR)

---

MSv36020V1
if (XFRC)
{
  Reset Error Count
  Mask ACK
  De-allocate Channel
}
else
if (STALL or FRMOR)
{
  Mask ACK
  Unmask CHH
  Disable Channel
  if (STALL)
  {
    Transfer Done = 1
  }
}
else
if (NAK or TXERR)
{
  Rewind Buffer Pointers
  Reset Error Count
  Mask ACK
  Unmask CHH
  Disable Channel
}
else
if (CHH)
{
  Mask CHH
  if (Transfer Done or (Error_count == 3))
  {
    De-allocate Channel
  }
  else
  {
    Re-initialize Channel (in next b_interval - 1 Frame)
  }
}
else
if (ACK)
{
  Reset Error Count
  Mask ACK
}
The application uses the NPTXFE interrupt in O TG_GINTSTS to find the transmit FIFO space.

Interrupt IN

Unmask (NAK/ TXERR/ XFRC/ BBERR/ STALL/ FRMOR/ DTERR)

if (XFRC)
{
    Reset Error Count
    Mask ACK
    if (OTG_HCTSIZx.PKTCNT == 0)
    {
        De-allocate Channel
    }
    else
    {
        Transfer Done = 1
        Unmask CHH
        Disable Channel
    }
}

else

if (STALL or FRMOR or NAK or DTERR or BBERR)
{
    Mask ACK
    Unmask CHH
    Disable Channel
    if (STALL or BBERR)
    {
        Reset Error Count
        Transfer Done = 1
    }
    else
    if (!FRMOR)
    {
        Reset Error Count
    }
    
}
else

if (TXERR)
{
    Increment Error Count
    Unmask ACK
    Unmask CHH
    Disable Channel
}
else
if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
        Re-initialize Channel (in next b_interval / Frame)
}
else
if (ACK)
{
    Reset Error Count
    Mask ACK
}

• **Interrupt IN transactions**
The assumptions are:
  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1024 bytes).
  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1031 bytes).
  – Periodic request queue depth = 4.

• **Normal interrupt IN operation**
The sequence of operations is as follows:
1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG_FS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG_FS host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-
initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.
1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in Figure 907. The assumptions are:
  - The application is attempting to send one packet every frame (up to 1 maximum
packet size), starting with an odd frame. (transfer size = 1 024 bytes).
– The periodic transmit FIFO can hold one packet (1 Kbyte).
– Periodic request queue depth = 4.

The sequence of operations is as follows:
1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_FS host writes an entry to the periodic request queue.
4. The OTG_FS host attempts to send the OUT token in the next frame (odd).
5. The OTG_FS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses
1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

  ```c
  Unmask (FRMOR/XFRC)
  if (XFRC)
  ```
De-allocate Channel
}
else
  if (FRMOR)
    {
      Unmask CHH
      Disable Channel
    }
else
  if (CHH)
    {
      Mask CHH
      De-allocate Channel
    }

Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
  {
    if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
      {
        Reset Error Count
        De-allocate Channel
      }
  else
    {
      Unmask CHH
      Disable Channel
    }
} else
if (TXERR or BBERR)
  {
    Increment Error Count
    Unmask CHH
    Disable Channel
  } else
if (CHH)
  {
    Mask CHH
    if (Transfer Done or (Error_count == 3))
      {
        De-allocate Channel
      }
else
{
    Re-initialize Channel
}

- **Isochronous IN transactions**
  
The assumptions are:
  - The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).
  - The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).
  - Periodic request queue depth = 4.

  The sequence of operations is as follows:
  1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
  2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
  3. The OTG_FS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
  4. The OTG_FS host attempts to send an IN token in the next odd frame.
  5. As soon as the IN packet is received and written to the receive FIFO, the OTG_FS host generates an RXFLVL interrupt.
  6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
  7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
  8. The core generates an XFRC interrupt as soon as the receive packet status is read.
  9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.
Figure 908. Isochronous IN transactions

1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**
  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic...
transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core’s periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**
  OTG_FS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

When OTG_FS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

When OTG_FS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

### 61.16.5 Device programming model

#### Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   - SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   - INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   - OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   - STUPM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DOEPMSK
   - XFRCM = 1 in OTG_DIEPMSK
   - TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   - Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   - Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   - STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
At this point, all initialization required to receive SETUP packets is done.

**Endpoint initialization on enumeration completion**

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

**Endpoint initialization on SetAddress command**

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

**Endpoint initialization on SetConfiguration/SetInterface command**

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

**Endpoint activation**

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.
1. Program the characteristics of the required endpoint into the following fields of the
   OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx
   register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number

2. Once the endpoint is activated, the core starts decoding the tokens addressed to that
   endpoint and sends out a valid handshake for each valid token received for the
   endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the
   OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx
   register (for OUT or bidirectional endpoints).

2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint,
   which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle
traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data
OUT transfers and SETUP transactions.

- Packet read

This section describes how to read packets (OUT data and SETUP packets) from the
receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read
   the receive status pop register (OTG_GRXSTSP).

2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to
   RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.

3. If the received packet’s byte count is not 0, the byte count amount of data is popped
   from the receive data FIFO and stored in memory. If the received packet byte count is
   0, no data is popped from the receive data FIFO.

4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
      PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),
      DPID = (0b00).
      These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
      PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,
DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.

c) Setup stage done pattern:
PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).
These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.

d) Data OUT packet pattern:
PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.

e) Data transfer completed pattern:
PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.

6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 909 provides a flowchart of the above procedure.

Figure 909. Receive FIFO packet read

SETUP transactions
This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

**Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.  
   - STUPCNT = 3 in OTG_DOEPTSIZx

2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   - The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   - 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.

4. The application must read and discard the setup stage done word from the receive FIFO.

**Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   - The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.

2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   - The first word contains control information used internally by the core
   - The second word contains the first 4 bytes of the SETUP command
   - The third word contains the last 4 bytes of the SETUP command

3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.

4. On the AHB side, SETUP packets are emptied by the application.

5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.

6. The core clears the endpoint enable bit for control OUT endpoints.

**Application programming sequence**
1. Program the OTG_DOEPTSIZx register.
   - STUPCNT = 3

2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.

3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   - On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 910. Processing a SETUP packet

- Handling more than three back-to-back SETUP packets

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_FS controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

- Setting the global OUT NAK

Internal data flow:
1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the
space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets.

2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.

3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).

4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL

2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.

3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).

4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register

5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL

6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:
1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   - SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   - EPDIS = 1 in OTG_DOEPCTLx
   - SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   - EPDIS = 0 in OTG_DOEPCTLx
   - EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   - SGONAK = 0 in OTG_DCTL

• **Transfer Stop Programming for OUT endpoints**

   The application must use the following programming sequence to stop any transfers (because of an interrupt from the host, typically a reset).

   **Sequence of operations:**
   1. Enable all OUT endpoints by setting
      - EPENA = 1 in all OTG_DOEPCTLx registers.
   2. Flush the RxFIFO as follows
      - Poll OTG_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
      - Perform read modify write operation on OTG_GRSTCTL.RXFFLSH = 1
      - Poll OTG_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
   3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “Setting the global OUT NAK on page 3034”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in OTG_DCTL
   4. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
   5. Disable all active OUT endpoints by programming the following register bits:
      - EPDIS = 1 in registers OTG_DOEPCTLx
      - SNAK = 1 in registers OTG_DOEPCTLx
   6. Wait for the EPDIS interrupt in OTG_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in OTG_DOEPINTx indicates that the
corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:
- EPENA = 0 in registers OTG_DOEPCTLx
- EPDIS = 0 in registers OTG_DOEPCTLx
- SNAK = 0 in registers OTG_DOEPCTLx

• **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:
1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint's transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   - transfer size[EPNUM] = \( n \times (MPSIZ[EPNUM] + 4 - (MPSIZ[EPNUM] \mod 4)) \)
   - packet count[EPNUM] = \( n \)
   - \( n > 0 \)
3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   - Payload size in memory = application programmed initial transfer size – core updated final transfer size
   - Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:
1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.

4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.

5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.

6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)

7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:
1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   - EPENA = 1 in OTG_DOEPCTLx
   - CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   - This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPI TX ) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:
1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:
1. The internal data flow for isochronous OUT endpoints is the same as that for non-
   isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and
   clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core
   receives data on an isochronous OUT endpoint in a particular frame only if the
   following condition is met:
   – $\text{EONUM (in OTG\_DOEPCTLx) = FNSOF}[0] \text{ (in OTG\_DSTS)}$
3. When the application completely reads an isochronous OUT data packet (data and
   status) from the receive FIFO, the core updates the RXDPID field in OTG\_DOEPTSIZx
   with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:
1. Program the OTG\_DOEPTSIZx register for the transfer size and the corresponding
   packet count
2. Program the OTG\_DOEPCTLx register with the endpoint characteristics and set the
   endpoint enable, ClearNAK, and Even/Odd frame bits.
   – $\text{EPENA = 1}$
   – $\text{CNAK = 1}$
   – $\text{EONUM = (0: Even/1: Odd)}$
3. Wait for the RXFLVL interrupt (in OTG\_GINTSTS) and empty the data packets from the
   receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG\_DOEPINTx) marks the completion of the
   isochronous OUT data transfer. This interrupt does not necessarily mean that the data
   in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the
   application can detect the INCOMPISOOUT interrupt in OTG\_GINTSTS.
6. Read the OTG\_DOEPTSIZx register to determine the size of the received transfer and
   to determine the validity of the data received in the frame. The application must treat
   the data received in memory as valid only if one of the following conditions is met:
   – $\text{RXDPID = DATA0 (in OTG\_DOEPTSIZx) and the number of USB packets in}
     \text{which this payload was received = 1}$
   – $\text{RXDPID = DATA1 (in OTG\_DOEPTSIZx) and the number of USB packets in}
     \text{which this payload was received = 2}$
   \text{The number of USB packets in which this payload was received =}
   \text{Application programmed initial packet count – core updated final packet count}
   \text{The application can discard invalid data packets.}

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data
packets are dropped inside the core.

Internal data flow:
1. For isochronous OUT endpoints, the XFRC interrupt (in OTG\_DOEPINTx) may not
   always be asserted. If the core drops isochronous OUT data packets, the application
could fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
– When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
– When the isochronous OUT data packet is received with CRC errors
– When the isochronous OUT token received by the core is corrupted
– When the application is very slow in reading the data from the receive FIFO

2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:
1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.

2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.

3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   – EPENA = 1 (in OTG_DOEPCTLx)

4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.

5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.

6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.
1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   - When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).
     The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the
   STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a
   SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must
   be set or cleared before the application sets up the status stage transfer on the control
   endpoint.

Examples
This section describes and depicts some fundamental transfer types and scenarios.
- Bulk OUT transaction

*Figure 911* depicts the reception of a single Bulk OUT data packet from the USB to the AHB
and describes the events involved in the process.

*Figure 911. Bulk OUT transaction*

After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints
by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable
XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.
1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

**IN data transfers**

- **Packet write**
  This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.
  1. The application can either choose the polling or the interrupt mode.
     - In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
     - In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
     - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
     - To write zero length packet, the application must not look at the FIFO space.
  2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

  Internal data flow:
1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint’s transmit FIFO.

2. Non-isochronous IN tokens receive a NAK handshake reply
   - Isochronous IN tokens receive a zero-data-length packet reply

3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.

4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

**Application programming sequence:**

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   - SNAK = 1 in OTG_DIEPCTLx

2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.

3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.

4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   - INEPNEM = 0 in OTG_DIEPMSK

5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).
   - CNAK = 1 in OTG_DIEPCTLx

6. If the application masked this interrupt earlier, it must be unmasked as follows:
   - INEPNEM = 1 in OTG_DIEPMSK

---

**IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

**Application programming sequence:**
1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   - SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   - EPDIS = 1 in OTG_DIEPCTLx
   - SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   - EPENA = 0 in OTG_DIEPCTLx
   - EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   - TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   - TXFFLSH in (OTG_GRSTCTL) = 1
   The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

   **Transfer Stop Programming for IN endpoints**

   The application must use the following programming sequence to stop any transfers (because of an interrupt from the host, typically a reset).

   **Sequence of operations:**
   1. Disable the IN endpoint by setting:
      - EPDIS = 1 in all OTG_DIEPCTLx registers
   2. Wait for the EPDIS interrupt in OTG_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
      - EPDIS = 0 in OTG_DIEPCTLx
      - EPENA = 0 in OTG_DIEPCTLx
   3. Flush the TxFIFO by programming the following bits:
      - TXFFLSH = 1 in OTG_GRSTCTL
      - TXFNUM = “FIFO number specific to endpoint” in OTG_GRSTCTL
   4. The application can start polling till TXFFLSH in OTG_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

   **Generic non-periodic IN data transfers**

   Application requirements:
1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.

2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
     \[
     \text{Transfer size}[\text{EPNUM}] = x \times \text{MPSIZ}[\text{EPNUM}] + \text{sp}
     \]
     If (sp > 0), then \(\text{packet count}[\text{EPNUM}] = x + 1\).
     Otherwise, \(\text{packet count}[\text{EPNUM}] = x\).
   - To transmit a single zero-length data packet:
     \(\text{Transfer size}[\text{EPNUM}] = 0\)
     \(\text{Packet count}[\text{EPNUM}] = 1\)
   - To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
     \(\text{First transfer: transfer size}[\text{EPNUM}] = x \times \text{MPSIZ}[\text{EPNUM}];\) \(\text{packet count} = n;\)
     \(\text{Second transfer: transfer size}[\text{EPNUM}] = 0;\) \(\text{packet count} = 1;\)

3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.

4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   - Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) \(\times \text{MPSIZ}[\text{EPNUM}]\)
   - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:
1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK
handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.

5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.

6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.

7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.

8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.

2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.

3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

App. 1

Generic periodic IN data transfers

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of Generic non-periodic IN data transfers on page 3044 also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To
transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

\[
\text{transfer size}[\text{EPNUM}] = x \times \text{MPSIZ}[\text{EPNUM}] + sp
\]

(where \( x \) is an integer \( \geq 0 \), and \( 0 \leq sp < \text{MPSIZ}[\text{EPNUM}] \))

If \( sp > 0 \), packet count[EPNUM] = \( x + 1 \)
Otherwise, packet count[EPNUM] = \( x \);

\[
\text{MCNT}[\text{EPNUM}] = \text{packet count}[\text{EPNUM}]
\]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:

– transfer size[EPNUM] = 0

packet count[EPNUM] = 1

\[
\text{MCNT}[\text{EPNUM}] = \text{packet count}[\text{EPNUM}]
\]

2. The application can only schedule data transfers one frame at a time.

– \((\text{MCNT} – 1) \times \text{MPSIZ} \leq \text{XFERSIZ} \leq \text{MCNT} \times \text{MPSIZ}\)

– \(\text{PKTCNT} = \text{MCNT} \) (in OTG\_DIEPTSZx)

– If \( \text{XFERSIZ} < \text{MCNT} \times \text{MPSIZ} \), the last data packet of the transfer is a short packet.

– Note that: \( \text{MCNT} \) is in OTG\_DIEPTSZx, \( \text{MPSIZ} \) is in OTG\_DIEPCTLx, \( \text{PKTCNT} \) is in OTG\_DIEPTSZx and \( \text{XFERSIZ} \) is in OTG\_DIEPTSZx

3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:

– A zero data length packet would be transmitted on the USB for isochronous IN endpoints

– A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.

2. The application must also write the required data to the associated transmit FIFO for the endpoint.

3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.

4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO
mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.

- A zero-length data packet is transmitted on the USB for isochronous IN endpoints
- A NAK handshake is transmitted on the USB for interrupt IN endpoints

5. The packet count for the endpoint is decremented by 1 under the following conditions:
- For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
- For interrupt endpoints, when an ACK handshake is transmitted
- When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.

6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:
1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- Incomplete isochronous IN data transfers

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:
1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it
eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.

3. The application must set the NAK bit and the disable bit for the endpoint.

4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).

2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.

3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.

4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.

5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   - SNAK = 1 in OTG_DIEPCTLx
   - EPDIS = 1 in OTG_DIEPCTLx

6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:
1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the
   application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on
   the endpoint type. In case of a non-periodic endpoint, the application must re-enable
   the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the
   STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a
   SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the
   STALL bit must be set or cleared before the application sets up the status stage
   transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host
sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the
application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt
in OTG_DOEPINTx during the data stage of the control transfer, after the core has
transferred the amount of data specified in the SETUP packet. Then, when the application
receives this interrupt, it must set the STALL bit in the corresponding endpoint control
register, and clear this interrupt.

61.16.6 Worst case response time

When the OTG_FS controller acts as a device, there is a worst case response time for any
tokens that follow an isochronous OUT. This worst case response time depends on the AHB
clock frequency.

The core registers are in the AHB domain, and the core does not accept another token
before updating these register values. The worst case is for any token following an
isochronous OUT, because for an isochronous transaction, there is no handshake and the
next token could come sooner. This worst case value is 7 PHY clocks when the AHB clock
is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK
and drops isochronous and SETUP tokens. The host interprets this as a timeout condition
for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete
isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer
interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were
dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY
clocks after it has received an IN token, to get the FIFO status, and thus the first data from
the PFC block. This time involves the synchronization delay between the PHY and AHB
clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock.
In this case, the delay is 5 clocks.
Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

*Figure 912* has the following signals:
- `tkn_rcvd`: Token received information from MAC to PFC
- `dynced_tkn_rcvd`: Doubled sync `tkn_rcvd`, from PCLK to HCLK domain
- `spr_read`: Read to SPRAM
- `spr_addr`: Address to SPRAM
- `spr_rdata`: Read data from SPRAM
- `srcbuf_push`: Push to the source buffer
- `srcbuf_rdata`: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 660: TRDT values*.

*Figure 912. TRDT max timing case*
61.16.7 OTG programming model

The OTG_FS controller is an OTG device. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device.
62 USB on-the-go high-speed (OTG_HS)

62.1 Introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG_HS controller.

The following acronyms are used throughout the section:

- FS Full-speed
- LS Low-speed
- HS High-speed
- MAC Media access controller
- OTG On-the-go
- PFC Packet FIFO controller
- PHY Physical layer
- USB Universal serial bus
- UTMU USB 2.0 Transceiver Macrocell interface (UTMI)
- LPM Link power management
- BCD Battery charging detector
- HNP Host negotiation protocol
- SRP Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the On-The-Go Supplement to the USB 2.0 Specification. It can also be configured as a host-only or device-only controller, fully compliant with the USB 2.0 Specification. OTG_HS supports the speeds defined in the Table 663: OTG_HS speeds supported below. The only external device required is a charge pump for VBUS in OTG mode.

<table>
<thead>
<tr>
<th></th>
<th>HS (480 Mbit/s)</th>
<th>FS (12 Mbit/s)</th>
<th>LS (1.5 Mbit/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host mode</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Device mode</td>
<td>X</td>
<td>X</td>
<td>-</td>
</tr>
</tbody>
</table>
62.2 OTG_HS main features

The main features can be divided into three categories: general, host-mode, and device-mode features.

62.2.1 General features

The OTG_HS interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0.
- OTG_HS supports the following PHY interfaces:
  - A UTMI interface for internal HS PHY
- It includes support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - It allows host to turn VBUS off to conserve battery power in OTG applications.
  - It supports OTG monitoring of VBUS levels with internal comparators.
- It is software-configurable to operate as:
  - USB On-The-Go Full-Speed Dual Role device
- It supports HS SOF and LS Keep-alives with
  - SOF pulse PAD connectivity
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- OTG_HS embeds an internal DMA with thresholding support and software selectable AHB burst type in DMA mode.
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 4 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets.
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2.
62.2.2 Host-mode features

The OTG_HS interface main features and requirements in host-mode are the following:

- External charge pump for VBUS voltage generation
- Up to 16 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 16 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 16 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

62.2.3 Peripheral-mode features

The OTG_HS interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 8 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 8 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 9 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

62.3 OTG_HS implementation

<table>
<thead>
<tr>
<th>USB features</th>
<th>OTG_HS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device bidirectional endpoints (including EP0)</td>
<td>9</td>
</tr>
<tr>
<td>Host mode channels</td>
<td>16</td>
</tr>
<tr>
<td>Size of dedicated SRAM</td>
<td>4 Kbytes</td>
</tr>
<tr>
<td>USB 2.0 link power management (LPM) support</td>
<td>X</td>
</tr>
<tr>
<td>OTG revision supported</td>
<td>2.0</td>
</tr>
<tr>
<td>Battery charging detection (BCD) support</td>
<td>X</td>
</tr>
<tr>
<td>Integrated PHY</td>
<td>HS</td>
</tr>
</tbody>
</table>

1. “X” = supported, “-” = not supported, “FS” = supported in FS mode, “HS” = supported in HS mode.
62.4 OTG_HS functional description

62.4.1 OTG_HS block diagram

Figure 913. OTG_HS high-speed block diagram

62.4.2 OTG_HS pin and internal signals

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HS_DP</td>
<td>Digital input/output</td>
<td>USB OTG D+ line</td>
</tr>
<tr>
<td>OTG_HS_DM</td>
<td>Digital input/output</td>
<td>USB OTG D- line</td>
</tr>
<tr>
<td>OTG_HS_ID</td>
<td>Digital input</td>
<td>USB OTG ID</td>
</tr>
<tr>
<td>OTG_HS_VBUS</td>
<td>Analog input</td>
<td>USB OTG VBUS</td>
</tr>
<tr>
<td>OTG_HS_SOF</td>
<td>Digital output</td>
<td>USB OTG Start Of Frame (visibility)</td>
</tr>
</tbody>
</table>
62.4.3 OTG_HS core

The OTG_HS receives the 60 MHz clock from the reset and clock controller (RCC). This is typically generated in the PLL associated with the HS PHY and enabled in the RCC. This clock is used for driving the 60 MHz domain at high-speed (480 Mbit/s) and must be enabled prior to configuring the OTG core.

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in Section 62.12: OTG_HS interrupts.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 4-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG_HS peripheral, the AHB frequency must be higher than 30 MHz.

62.4.4 OTG detections

Additionally the OTG_HS uses the following functions:

- Integrated ID pull-up resistor used to sample the ID line for A/B device identification.
- $V_{BUS}$ sensing comparators with hysteresis used to detect $V_{BUS}$ valid, A-B session valid and session-end voltage thresholds. They are used to detect valid startup and end-of-session conditions, and constantly monitor the $V_{BUS}$ supply during USB operations.

62.4.5 High-speed OTG PHY connected to OTG_HS

Note: Refer to implementation table to determine if an HS PHY is embedded.

The USB OTG core includes an internal UTMI interface which is connected to the embedded HS PHY (see Section 62.4.1: OTG_HS block diagram).

<table>
<thead>
<tr>
<th>Signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>usb_sof</td>
<td>Digital output</td>
<td>USB OTG start-of-frame event for on chip peripherals</td>
</tr>
<tr>
<td>usb_wkup</td>
<td>Digital output</td>
<td>USB OTG wake-up event output</td>
</tr>
<tr>
<td>usb_gbl_it</td>
<td>Digital output</td>
<td>USB OTG global interrupt</td>
</tr>
</tbody>
</table>
62.5 OTG_HS dual role device (DRD)

![Diagram](MSv36917V2)

1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

62.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG_HS complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG_HS issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG_HS complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

62.6 OTG_HS as a USB peripheral

This section gives the functional description of the OTG_HS in the USB peripheral mode. The OTG_HS works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable.
- Peripheral only
  - The force device mode bit (FDMOD) in the Section 62.14.4: OTG USB configuration register (OTG_GUSBCFG) is set to 1, forcing the OTG_HS core to
work as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from \( V_{BUS} \).

**Figure 915. OTG_HS peripheral-only connection**

1. Use a regulator to build a bus-powered device.

### 62.6.1 Peripheral states

#### Powered state

The \( V_{BUS} \) input detects the B-session valid voltage by which the USB peripheral is allowed to enter the powered state (see USB2.0 section 9.1). The OTG_HS then automatically connects the DP pull-up resistor to signal full-speed device connection to the host and generates the session request interrupt (SRQINT bit in OTG_GINTSTS) to notify the powered state.

The \( V_{BUS} \) input also ensures that valid \( V_{BUS} \) levels are supplied by the host during USB operations. If a drop in \( V_{BUS} \) below B-session valid happens to be detected (for instance because of a power disturbance or if the host port has been switched off), the OTG_HS automatically disconnects and the session end detected (SEDET bit in OTG_GOTGINT) interrupt is generated to notify that the OTG_HS has exited the powered state.

In the powered state, the OTG_HS expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG_HS enters the Default state.

#### Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS
bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

**Default state**

In the Default state the OTG_HS expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG_HS then enters the address state and is ready to answer host transactions at the configured USB address.

**Suspended state**

The OTG_HS peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG_HS enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wake-up signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

### 62.6.2 Peripheral endpoints

The OTG_HS core instantiates the following USB endpoints:

- **Control endpoint 0:**
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Dedicated control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints

- **eight IN endpoints**
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has dedicated control (OTG_DIEPCTLx), transfer configuration (OTG_DIEPTSIZx), and status-interrupt (OTG_DIEPINTx) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on
which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

- eight OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a dedicated control (OTG_DOEPCTLx), transfer configuration (OTG_DOEPTSIZx) and status-interrupt (OTG_DOEPINTx) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

**Endpoint control**

- The following endpoint controls are available to the application through the device endpoint-x IN/OUT control register (OTG_DIEPCTLx/OTG_DOEPCTLx):
  - endpoint enable/disable
  - endpoint activate in current configuration
  - program USB transfer type (isochronous, bulk, interrupt)
  - program supported packet size
  - program Tx FIFO number associated with the IN endpoint
  - program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - optionally program the STALL bit to always stall host tokens to that endpoint
  - optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

**Endpoint transfer**

The device endpoint-x transfer size registers (OTG_DIEPTSIZx/OTG_DOEPTSIZx) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG_HS core updates them with the current transfer status.

The following transfer parameters can be programmed:

- transfer size in bytes
- number of packets that constitute the overall transfer size

**Endpoint status/interrupt**

The device endpoint-x interrupt registers (OTG_DIEPINTx/OTG_DOPEPINTx) indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in
the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINTE) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINTE and OTG_GINTSTS registers.

The peripheral core provides the following status checks and interrupt generation:

- transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- setup stage has been done (control-out only)
- associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- babble error condition has been detected
- endpoint disable by application is effective
- endpoint NAK by application is effective (isochronous-in only)
- more than 3 back-to-back setup packets were received (control-out only)
- timeout condition detected (control-in only)
- isochronous out packet has been dropped, without generating an interrupt

### 62.7 OTG_HS as a USB host

This section gives the functional description of the OTG_HS in the USB host mode. The OTG_HS works as a USB host in the following circumstances:

- OTG A-host
  - OTG A-device default state when the A-side of the USB cable is plugged in
- A-device
  - If the ID line is present, functional and connected to the A-side of the USB cable. Integrated pull-down resistors are automatically set on the DP/DM lines.
- Host only
  - The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG_HS core to work as a USB host-only. Integrated pull-down resistors are automatically set on the DP/DM lines.

**Note:** *On-chip 5 V VBUS generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive the 5 V VBUS line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.*
1. $V_{DD}$ range is between 2 V and 3.6 V.

### 62.7.1 USB host states

**Host port power**

On-chip 5 V $V_{BUS}$ generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V $V_{BUS}$ line. The external charge pump can be driven by any GPIO output or via an I2C interface connected to an external PMIC (power management IC). When the application decides to power on $V_{BUS}$, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

**$V_{BUS}$ valid**

In Host mode, the $V_{BUS}$ sensing pin does not need to be connected to $V_{BUS}$.

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the $V_{BUS}$ generation and clear the port power bit.

**Host detection of a peripheral connection**

USB peripherals or B-device are detected as soon as they are connected. The OTG_HS core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

**Host detection of peripheral a disconnection**

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

**Host enumeration**

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for
a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

**Host suspend**

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG_HS core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device’s initiative (remote wake-up). In this case the remote wake-up interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wake-up signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

### 62.7.2 Host channels

The OTG_HS core instantiates 16 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 16 transfer requests at the same time. If more than 16 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of dedicated control (OTG_HCCHARx), transfer configuration (OTG_HCTSIZEx) and status/interrupt (OTG_HCINTx) registers with associated mask (OTG_HCINTMSKx) registers.

**Host channel control**

- The following host channel controls are available to the application through the host channel-x characteristics register (OTG_HCCHARx):
  - Channel enable/disable
  - Program the HS/FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames
Host channel transfer

The host channel transfer size registers (OTG_HCTSIZx) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG_HS core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-x interrupt register (OTG_HCINTx) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers. The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

62.7.3 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the
requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG_HS core is fully responsible for the management of the periodic and nonperiodic request queues. The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of eight entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of eight pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

### 62.8 OTG_HS SOF trigger

**Figure 917. SOF connectivity (SOF trigger output to TIM and ITR1 connection)**

The OTG_HS core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.
62.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (HS/FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

62.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated. The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

62.9 OTG_HS low-power modes

Table 667 below defines the STM32 low power modes and their compatibility with the OTG.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
<th>USB compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run</td>
<td>MCU fully active</td>
<td>Required when USB not in suspend state.</td>
</tr>
<tr>
<td>Sleep</td>
<td>USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.</td>
<td>Available while USB is in suspend state.</td>
</tr>
<tr>
<td>Stop</td>
<td>USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).</td>
<td>Available while USB is in suspend state.</td>
</tr>
<tr>
<td>Standby</td>
<td>Powered-down. The peripheral must be reinitialized after exiting Standby mode.</td>
<td>Not compatible with USB applications.</td>
</tr>
</tbody>
</table>

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to Section 6: Power control (PWR) to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.
The power consumption of the OTG PHY is controlled by the following bit in the general core configuration register:

- **VBUS detection enable (OTG_GCCFG/VBDEN)**
  It switches on/off the VBUS sensing comparators associated with OTG operations.

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- **Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)**
  When setting the stop PHY clock bit in the clock gating control register, most of the transceiver is disabled, and only the part in charge of detecting the asynchronous resume or remote wake-up event is kept alive.

- **Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)**
  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG_HS core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.

- **USB system stop**
  When the OTG_HS is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).
  The OTG_HS core automatically reactivates both system and USB clocks by asynchronous detection of remote wake-up (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG_HS core.

### 62.10 OTG_HS Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in Figure 918.

For a dynamic update, it is required to set RLDCTRL=1.
62.11 OTG_HS data FIFOs

The USB system features 4 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG_HS core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device’s role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.
62.11.1 Peripheral FIFO architecture

Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG_HS core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.
Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint 0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

62.11.2 Host FIFO architecture

Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG_HS core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.
Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG_HS core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG_HS core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXSELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG_HS core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

62.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application must allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of \((\text{largest packet size} / 4) \times 1\) must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two \((\text{largest packet size} / 4) + 1\) spaces must be allocated to receive back-to-back packets. Typically, two \((\text{largest packet size} / 4) + 1\) spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.
Device RxFIFO =
(5 * number of control endpoints + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = (5 * 1 + 8) + ((1,024 / 4) +1) + (2 * 4) + 1 = 279

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:
Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = (largest USB packet used / 4) + 1 for status information + 1 transfer complete

Example: Host RxFIFO = ((1,024 / 4) + 1) + 1 = 258

Transmit FIFO RAM allocation:
The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = (512 / 4) = 128

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = (1,024 / 4) = 256

Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.
62.12 OTG_HS interrupts

When the OTG_HS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

*Figure 921* shows the interrupt hierarchy.
1. OTG_HS_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.
62.13 OTG_HS control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG_HS controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG_HS registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG_HS controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

62.13.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Address offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GOTGCTL</td>
<td>0x000</td>
<td>Section 62.14.1: OTG control and status register (OTG_GOTGCTL)</td>
</tr>
<tr>
<td>OTG_GOTGINT</td>
<td>0x004</td>
<td>Section 62.14.2: OTG interrupt register (OTG_GOTGINT)</td>
</tr>
<tr>
<td>OTG_GAHBCFG</td>
<td>0x008</td>
<td>Section 62.14.3: OTG AHB configuration register (OTG_GAHBCFG)</td>
</tr>
<tr>
<td>OTG_GUSBCFG</td>
<td>0x00C</td>
<td>Section 62.14.4: OTG USB configuration register (OTG_GUSBCFG)</td>
</tr>
<tr>
<td>OTG_GRSTCTL</td>
<td>0x010</td>
<td>Section 62.14.5: OTG reset register (OTG_GRSTCTL)</td>
</tr>
<tr>
<td>OTG_GINTSTS</td>
<td>0x014</td>
<td>Section 62.14.6: OTG core interrupt register [alternate] (OTG_GINTSTS) Section 62.14.7: OTG core interrupt register [alternate] (OTG_GINTSTS)</td>
</tr>
</tbody>
</table>
### Table 668. Core global control and status registers (CSRs) (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Address offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GINTMSK</td>
<td>0x018</td>
<td>Section 62.14.8: OTG interrupt mask register [alternate] (OTG_GINTMSK)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.9: OTG interrupt mask register [alternate] (OTG_GINTMSK)</td>
</tr>
<tr>
<td>OTG_GRXSTSR</td>
<td>0x01C</td>
<td>Section 62.14.10: OTG receive status debug read register [alternate] (OTG_GRXSTSR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.11: OTG receive status debug read register [alternate] (OTG_GRXSTSR)</td>
</tr>
<tr>
<td>OTG_GRXSTSP</td>
<td>0x020</td>
<td>Section 62.14.12: OTG status read and pop registers (OTG_GRXSTSP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.13: OTG status read and pop registers [alternate] (OTG_GRXSTSP)</td>
</tr>
<tr>
<td>OTG_GRXFSIZ</td>
<td>0x024</td>
<td>Section 62.14.14: OTG receive FIFO size register (OTG_GRXFSIZ)</td>
</tr>
<tr>
<td>OTG_HNPTXFSIZ/OTG_DIEPTXF0(1)</td>
<td>0x028</td>
<td>Section 62.14.15: OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.16: Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)</td>
</tr>
<tr>
<td>OTG_HNPTXSTS</td>
<td>0x02C</td>
<td>Section 62.14.17: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)</td>
</tr>
<tr>
<td>OTG_GCCFG</td>
<td>0x038</td>
<td>Section 62.14.18: OTG general core configuration register (OTG_GCCFG)</td>
</tr>
<tr>
<td>OTG_CID</td>
<td>0x03C</td>
<td>Section 62.14.19: OTG core ID register (OTG_CID)</td>
</tr>
<tr>
<td>OTG_GLPMCFG</td>
<td>0x54</td>
<td>Section 62.14.20: OTG core LPM configuration register (OTG_GLPMCFG)</td>
</tr>
<tr>
<td>OTG_HPTXFSIZ</td>
<td>0x100</td>
<td>Section 62.14.21: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)</td>
</tr>
<tr>
<td>OTG_DIEPTXFx</td>
<td>0x104</td>
<td>Section 62.14.22: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)</td>
</tr>
<tr>
<td></td>
<td>0x108</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x120</td>
<td></td>
</tr>
</tbody>
</table>

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

**Host-mode CSR map**

These registers must be programmed every time the core changes to host mode.

### Table 669. Host-mode control and status registers (CSRs)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HCFG</td>
<td>0x400</td>
<td>Section 62.14.24: OTG host configuration register (OTG_HCFG)</td>
</tr>
<tr>
<td>OTG_HFIR</td>
<td>0x404</td>
<td>Section 62.14.25: OTG host frame interval register (OTG_HFIR)</td>
</tr>
<tr>
<td>OTG_HFNUM</td>
<td>0x408</td>
<td>Section 62.14.26: OTG host frame number/frame time remaining register (OTG_HFNUM)</td>
</tr>
<tr>
<td>OTG_HPTXSTS</td>
<td>0x410</td>
<td>Section 62.14.27: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)</td>
</tr>
</tbody>
</table>
### Table 669. Host-mode control and status registers (CSRs) (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HAINT</td>
<td>0x414</td>
<td>Section 62.14.28: OTG host all channels interrupt register (OTG_HAINT)</td>
</tr>
<tr>
<td>OTG_HAINTMSK</td>
<td>0x418</td>
<td>Section 62.14.29: OTG host all channels interrupt mask register (OTG_HAINTMSK)</td>
</tr>
<tr>
<td>OTG_HPRT</td>
<td>0x440</td>
<td>Section 62.14.30: OTG host port control and status register (OTG_HPRT)</td>
</tr>
<tr>
<td>OTG_HCCHARRx</td>
<td>0x500, 0x520</td>
<td>Section 62.14.31: OTG host channel x characteristics register (OTG_HCCHARRx)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x6E0</td>
<td></td>
</tr>
<tr>
<td>OTG_HCSPLTx</td>
<td>0x504, 0x524</td>
<td>Section 62.14.32: OTG host channel x split control register (OTG_HCSPLTx)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x6E4</td>
<td></td>
</tr>
<tr>
<td>OTG_HCINTx</td>
<td>0x508, 0x528</td>
<td>Section 62.14.33: OTG host channel x interrupt register (OTG_HCINTx)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x6E8</td>
<td></td>
</tr>
<tr>
<td>OTG_HCINTMSKx</td>
<td>0x50C, 0x52C</td>
<td>Section 62.14.34: OTG host channel x interrupt mask register (OTG_HCINTMSKx)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x6EC</td>
<td></td>
</tr>
<tr>
<td>OTG_HCTSIZx</td>
<td>0x510, 0x530</td>
<td>Section 62.14.35: OTG host channel x transfer size register (OTG_HCTSIZx)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x6F0</td>
<td></td>
</tr>
<tr>
<td>OTG_HCDMAx</td>
<td>0x514, 0x534</td>
<td>Section 62.14.36: OTG host channel x DMA address register (OTG_HCDMAx)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0x6F4</td>
<td></td>
</tr>
</tbody>
</table>

### Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

### Table 670. Device-mode control and status registers

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_DCFG</td>
<td>0x800</td>
<td>Section 62.14.38: OTG device configuration register (OTG_DCFG)</td>
</tr>
<tr>
<td>OTG_DCTL</td>
<td>0x804</td>
<td>Section 62.14.39: OTG device control register (OTG_DCTL)</td>
</tr>
<tr>
<td>OTG_DSTS</td>
<td>0x808</td>
<td>Section 62.14.40: OTG device status register (OTG_DSTS)</td>
</tr>
</tbody>
</table>
Table 670. Device-mode control and status registers (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_DIEPMSK</td>
<td>0x810</td>
<td>Section 62.14.41: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)</td>
</tr>
<tr>
<td>OTG_DOEPMASK</td>
<td>0x814</td>
<td>Section 62.14.42: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMASK)</td>
</tr>
<tr>
<td>OTG_DAINT</td>
<td>0x818</td>
<td>Section 62.14.43: OTG device all endpoints interrupt register (OTG_DAINT)</td>
</tr>
<tr>
<td>OTG_DAINTMSK</td>
<td>0x81C</td>
<td>Section 62.14.44: OTG all endpoints interrupt mask register (OTG_DAINTMSK)</td>
</tr>
<tr>
<td>OTG_DTHRCTL</td>
<td>0x830</td>
<td>Section 62.14.45: OTG device threshold control register (OTG_DTHRCTL)</td>
</tr>
<tr>
<td>OTG_DIEPEMPMSK</td>
<td>0x834</td>
<td>Section 62.14.46: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)</td>
</tr>
<tr>
<td>OTG_DIEPCTLx</td>
<td>0x900, 0x920, ... 0xA00</td>
<td>Section 62.14.47: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) Section 62.14.48: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)</td>
</tr>
<tr>
<td>OTG_DIEPINTx</td>
<td>0x908, 0x928, ... 0x9E8</td>
<td>Section 62.14.49: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)</td>
</tr>
<tr>
<td>OTG_DIEPTSIZ0</td>
<td>0x910</td>
<td>Section 62.14.50: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)</td>
</tr>
<tr>
<td>OTG_DIEPDMAx</td>
<td>0x914, 0x934, ... 0x9F4</td>
<td>Section 62.14.51: OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)</td>
</tr>
<tr>
<td>OTG_DTXFSTSx</td>
<td>0x918, 0x938, ..... 0x9F8</td>
<td>Section 62.14.52: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)</td>
</tr>
<tr>
<td>OTG_DIEPTSIZx</td>
<td>0x930, 0x950, ... 0x9F0</td>
<td>Section 62.14.53: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)</td>
</tr>
<tr>
<td>OTG_DOEPCTL0</td>
<td>0xB00</td>
<td>Section 62.14.54: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)</td>
</tr>
<tr>
<td>OTG_DOEPINTx</td>
<td>0xB08, 0xB28, ... 0xC08</td>
<td>Section 62.14.55: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)</td>
</tr>
</tbody>
</table>
Table 670. Device-mode control and status registers (continued)

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_DOEPTSIZ0</td>
<td>0xB10</td>
<td>Section 62.14.56: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)</td>
</tr>
<tr>
<td>OTG_DOEPDMAx</td>
<td>0xB14 0xB34 ... 0xC14</td>
<td>Section 62.14.57: OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)</td>
</tr>
<tr>
<td>OTG_DOEPCTLx</td>
<td>0xB20 0xB40 ... 0xC00</td>
<td>Section 62.14.58: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Section 62.14.59: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)</td>
</tr>
<tr>
<td>OTG_DOEPTSIZx</td>
<td>0xB30 0xB50 .. 0xBF0</td>
<td>Section 62.14.60: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)</td>
</tr>
</tbody>
</table>

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 671. Data FIFO (DFIFO) access register map

<table>
<thead>
<tr>
<th>FIFO access register section</th>
<th>Offset address</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device IN endpoint 0/Host OUT Channel 0: DFIFO write access</td>
<td>0x1000–0x1FFC</td>
<td>w r</td>
</tr>
<tr>
<td>Device OUT endpoint 0/Host IN Channel 0: DFIFO read access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device IN endpoint 1/Host OUT Channel 1: DFIFO write access</td>
<td>0x2000–0x2FFC</td>
<td>w r</td>
</tr>
<tr>
<td>Device OUT endpoint 1/Host IN Channel 1: DFIFO read access</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Device IN endpoint $x^{(1)}$/Host OUT Channel $x^{(1)}$: DFIFO write access</td>
<td>0xX000–0xXFFC</td>
<td>w r</td>
</tr>
<tr>
<td>Device OUT endpoint $x^{(1)}$/Host IN Channel $x^{(1)}$: DFIFO read access</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Where $x$ is 8 in device mode and 15 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.
62.14 OTG_HS registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

62.14.1 OTG control and status register (OTG_GOTGCTL)

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

Address offset: 0x000

Reset value: 0x0001 0000

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_GOTGCTL</td>
<td>0xE00–0xE04</td>
<td>Section 62.14.61: OTG power and clock gating control register (OTG_PCGCCTL)</td>
</tr>
</tbody>
</table>

Table 672. Power and clock gating control and status registers

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Offset address</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_PCGCCTL</td>
<td>0xE00–0xE04</td>
<td>Section 62.14.61: OTG power and clock gating control register (OTG_PCGCCTL)</td>
</tr>
</tbody>
</table>

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

Bit 21 CURMOD: Current mode of operation
Indicates the current mode (host or device).
0: Device mode
1: Host mode

Bit 20 OTGVER: OTG version
Selects the OTG revision.
0: OTG Version 1.3. OTG1.3 is obsolete for new product development.
1: OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 BSVLD: B-session valid
Indicates the device mode transceiver status.
0: B-session is not valid.
1: B-session is valid.
In OTG mode, the user can use this bit to determine if the device is connected or disconnected.

Note: Only accessible in device mode.
Bit 18 **ASVLD**: A-session valid
   Indicates the host mode transceiver status.
   0: A-session is not valid
   1: A-session is valid
   *Note*: **Only accessible in host mode.**

Bit 17 **DBCT**: Long/short debounce time
   Indicates the debounce time of a detected connection.
   0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
   1: Short debounce time, used for soft connections (2.5 µs)
   *Note*: **Only accessible in host mode.**

Bit 16 **CIDSTS**: Connector ID status
   Indicates the connector ID status on a connect event.
   0: The OTG_HS controller is in A-device mode
   1: The OTG_HS controller is in B-device mode
   *Note*: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN**: Embedded host enable
   It is used to select between OTG A device state machine and embedded host state machine.
   0: OTG A device state machine is selected
   1: Embedded host state machine is selected

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **BVALOVAL**: B-peripheral session valid override value
   This bit is used to set override value for Bvalid signal when BVALOEN bit is set.
   0: Bvalid value is ‘0’ when BVALOEN = 1
   1: Bvalid value is ‘1’ when BVALOEN = 1
   *Note*: **Only accessible in device mode.**

Bit 6 **BVALOEN**: B-peripheral session valid override enable
   This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.
   0: Override is disabled and Bvalid signal from the respective PHY selected is used internally by the core
   1: Internally Bvalid received from the PHY is overridden with BVALOVAL bit value
   *Note*: **Only accessible in device mode.**

Bit 5 **AVALOVAL**: A-peripheral session valid override value
   This bit is used to set override value for Avalid signal when AVALOEN bit is set.
   0: Avalid value is ‘0’ when AVALOEN = 1
   1: Avalid value is ‘1’ when AVALOEN = 1
   *Note*: **Only accessible in host mode.**

Bit 4 **AVALOEN**: A-peripheral session valid override enable
   This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.
   0: Override is disabled and Avalid signal from the respective PHY selected is used internally by the core
   1: Internally Avalid received from the PHY is overridden with AVALOVAL bit value
   *Note*: **Only accessible in host mode.**
Bit 3 **VBVALOVAL**: $V_{BUS}$ valid override value
This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.
0: vbusvalid value is '0' when VBVALOEN = 1
1: vbusvalid value is '1' when VBVALOEN = 1
**Note**: Only accessible in host mode.

Bit 2 **VBVALOEN**: $V_{BUS}$ valid override enable
This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.
0: Override is disabled and vbusvalid signal from the respective PHY selected is used internally by the core.
1: Internally vbusvalid received from the PHY is overridden with VBVALOVAL bit value
**Note**: Only accessible in host mode.

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

### 62.14.2 OTG interrupt register (OTG_GOTGINT)

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Address offset: 0x004
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

```

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **ADTOCHG**: A-device timeout change
The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

**Note**: Accessible in both device and host modes.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **SEDET**: Session end detected
The core sets this bit to indicate that the level of the voltage on $V_{BUS}$ is no longer valid for a B-Peripheral session when $V_{BUS} < 0.8$ V.

**Note**: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.
62.14.3 OTG AHB configuration register (OTG_GAHBCFG)

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Address offset: 0x008
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>PTXFE LVL</th>
<th>TXFE LVL</th>
<th>DMAE N</th>
<th>HBSTLEN[3:0]</th>
<th>GINT MSK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFE LVL**: Periodic Tx FIFO empty level
Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered:
0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty
1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

*Note: Only accessible in host mode.*

Bit 7 **TXFE LVL**: Tx FIFO empty level

**Condition: device mode**
This bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:
0: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty
1: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

**Condition: host mode**
This bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:
0: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty
1: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bit 6 Reserved, must be kept at reset value.
Bit 5 **DMAEN**: DMA enabled
   0: The core operates in slave mode
   1: The core operates in DMA mode

Bits 4:1 **HBSTLEN[3:0]**: Burst length/type
   0000 Single: Bus transactions use single 32 bit accesses (not recommended)
   0001 INCR: Bus transactions use unspecified length accesses (not recommended, uses the
               INCR AHB bus command)
   0011 INCR4: Bus transactions target 4x 32 bit accesses
   0101 INCR8: Bus transactions target 8x 32 bit accesses
   0111 INCR16: Bus transactions based on 16x 32 bit accesses
   Others: Reserved

Bit 0 **GINTMSK**: Global interrupt mask
   The application uses this bit to mask or unmask the interrupt line assertion to itself.
   Irrespective of this bit setting, the interrupt status registers are updated by the core.
   0: Mask the interrupt assertion to the application.
   1: Unmask the interrupt assertion to the application.

*Note: Accessible in both device and host modes.*

### 62.14.4 OTG USB configuration register (OTG_GUSBCFG)

This register can be used to configure the core after power-on or a changing to host mode
or device mode. It contains USB and USB-PHY related configuration parameters. The
application must program this register before starting any transactions on either the AHB or
the USB. Do not make changes to this register after the initial programming.

Address offset: 0x00C

Reset value: 0x0000 1400

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bit 31  Reserved, must be kept at reset value.

Bit 30  **FDMOD**: Force device mode
        Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
        0: Normal mode
        1: Force device mode
        After setting the force bit, the application must wait at least 25 ms before the change takes effect.
        
        *Note:* Accessible in both device and host modes.

Bit 29  **FHMOD**: Force host mode
        Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
        0: Normal mode
        1: Force host mode
        After setting the force bit, the application must wait at least 25 ms before the change takes effect.
        
        *Note:* Accessible in both device and host modes.

Bits 28:26  Reserved, must be kept at reset value.

Bits 25:23  Reserved, must be kept at reset value.

Bit 22  **TSDPS**: TermSel DLine pulsing selection
        This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).
        0: Data line pulsing using utmi_txvalid (default)
        1: Data line pulsing using utmi_termselect

Bits 21:16  Reserved, must be kept at reset value.

Bit 15  **PHYLPC**: PHY Low-power clock select
        This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.
        0: 480 MHz internal PLL clock
        1: 48 MHz external clock
        In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14  Reserved, must be kept at reset value.

Bits 13:10  **TRDT[3:0]**: USB turnaround time
        These bits allow to set the turnaround time in PHY clocks. They must be configured according to Table 673: **TRDT values**, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.
        
        *Note:* Only accessible in device mode.

Bits 9:8  Reserved, must be kept at reset value.

Bit 7  Reserved, must be kept at reset value.

Bit 6  Reserved, must be kept at reset value.

Bit 5  Reserved, must be kept at reset value.
Bit 4  Reserved, must be kept at reset value.
Bit 3  Reserved, must be kept at reset value.

Bits 2:0  **TOCAL[2:0]:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

### Table 673. TRDT values

<table>
<thead>
<tr>
<th>AHB frequency range (MHz)</th>
<th>TRDT minimum value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>Max</td>
</tr>
<tr>
<td>30</td>
<td>-</td>
</tr>
<tr>
<td>0x9</td>
<td></td>
</tr>
</tbody>
</table>

#### 62.14.5 OTG reset register (OTG_GRSTCTL)

The application uses this register to reset various hardware features inside the core.

Address offset: 0x010

Reset value: 0x8000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td>rs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXFNUM[4:0]</td>
<td>TXF</td>
<td>RXF</td>
<td>FLUSH</td>
<td>FLUSH</td>
<td>FCST</td>
<td>PSRST</td>
<td>CSRST</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

*Note:* Accessible in both device and host modes.

Bit 30  **DMAREQ:** DMA request signal enabled

This bit indicates that the DMA request is in progress. Used for debug.

Bits 29:11  Reserved, must be kept at reset value.
Bits 10:6 **TXFNUM[4:0]**: Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

Condition: host mode

- 00000: Non-periodic Tx FIFO flush
- 00001: Periodic Tx FIFO flush
- 10000: Flush all the transmit FIFOs

Condition: device mode

- 00000: Tx FIFO 0 flush
- 00001: Tx FIFO 1 flush
- 00010: Tx FIFO 2 flush
- ...
- 01111: Tx FIFO 15 flush
- 10000: Flush all the transmit FIFOs

*Note:* Accessible in both device and host modes.

Bit 5 **TXFFLSH**: Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

- Read—NAK Effective interrupt ensures the core is not reading from the FIFO
- Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

*Note:* Accessible in both device and host modes.

Bit 4 **RXFFLSH**: Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

*Note:* Accessible in both device and host modes.

Bit 3 **Reserved, must be kept at reset value.**
Bit 2  **FCRST**: Host frame counter reset
The application writes this bit to reset the (micro-)frame number counter inside the core. 
When the (micro-)frame counter is reset, the subsequent SOF sent out by the core has a 
frame number of 0. 
When application writes "1" to the bit, it might not be able to read back the value as it gets 
cleared by the core in a few clock cycles.
*Note:*  *Only accessible in host mode.*

Bit 1  **PSRST**: Partial soft reset
Resets the internal state machines but keeps the enumeration info. Can be used to recover 
some specific PHY errors.
*Note:*  *Accessible in both device and host modes.*

Bit 0  **CSRST**: Core soft reset
Resets the HCLK and PHY clock domains as follows:
Clears the interrupts and all the CSR register bits except for the following bits:
– GATEHCLK bit in OTG_PCGCCTL
– STPPCLK bit in OTG_PCGCCTL
– FSLSPCS bits in OTG_HCFG
– DSPD bit in OTG_DCFG
– SDIS bit in OTG_DCTL
– OTG_GCCFG register
– FDMOD bit in OTG_GUSBCFG
– FHMOD bit in OTG_GUSBCFG
– PHYLPC bit in OTG_GUSBCFG
– TSDPS bit in OTG_GUSBCFG
All module state machines (except for the AHB slave unit) are reset to the Idle state, and all 
the transmit FIFOs and the receive FIFO are flushed.
Any transactions on the AHB Master are terminated as soon as possible, after completing the 
last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. 
The application can write to this bit any time it wants to reset the core. This is a self-clearing 
bit and the core clears this bit after all the necessary logic is reset in the core, which can take 
several clocks, depending on the current state of the core. Once this bit has been cleared, 
the software must wait at least 3 PHY clocks before accessing the PHY domain 
(synchronization delay). The software must also check that bit 31 in this register is set to 1 
(AHB Master is Idle) before starting any operation.
Typically, the software reset is used during software development and also when the user 
dynamically changes the PHY selection bits in the above listed USB configuration registers. 
When the user changes the PHY, the corresponding clock for the PHY is selected and used 
in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper 
operation.
*Note:*  *Accessible in both device and host modes.*
62.14.6  OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Host mode, see next section for Device mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>WKUPINT: Resume/remote wake-up detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Wake-up interrupt during suspend(L2) or LPM(L1) state.</td>
</tr>
<tr>
<td></td>
<td>– During suspend(L2):</td>
</tr>
<tr>
<td></td>
<td>In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.</td>
</tr>
<tr>
<td></td>
<td>– During LPM(L1):</td>
</tr>
<tr>
<td></td>
<td>This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.</td>
</tr>
<tr>
<td></td>
<td>Note: Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>SRQINT: Session request/new session detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In host mode, this interrupt is asserted when a session request is detected from the device.</td>
</tr>
<tr>
<td></td>
<td>In device mode, this interrupt is asserted when VBUS is in the valid range for a B-peripheral device. Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>DISCINT: Disconnect detected interrupt</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Asserted when a device disconnect is detected.</td>
</tr>
<tr>
<td></td>
<td>Note: Only accessible in host mode.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>CIDSCHG: Connector ID status change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The core sets this bit when there is a change in connector ID status.</td>
</tr>
<tr>
<td></td>
<td>Note: Accessible in both device and host modes.</td>
</tr>
</tbody>
</table>
Bit 27  **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLCFG).

This field is valid only if the LPMEN bit in OTG_GLCFG is set to 1.

Bit 26  **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

*Note:* Only accessible in host mode.

Bit 25  **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAIN register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

*Note:* Only accessible in host mode.

Bit 24  **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG_HS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

*Note:* Only accessible in host mode.

Bit 23  **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

*Note:* Only accessible in device mode.

Bit 22  **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake if the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN handshake.

Bit 21  **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.
Bit 20  **ISOIXFR**: Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

*Note: Only accessible in device mode.*

Bit 19  **OEPINT**: OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

*Note: Only accessible in device mode.*

Bit 18  **IEPINT**: IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

*Note: Only accessible in device mode.*

Bits 17:16  Reserved, must be kept at reset value.

Bit 15  **EOPF**: End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

*Note: Only accessible in device mode.*

Bit 14  **ISOODRP**: Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

*Note: Only accessible in device mode.*

Bit 13  **ENUMDNE**: Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

*Note: Only accessible in device mode.*

Bit 12  **USBRST**: USB reset

The core sets this bit to indicate that a reset is detected on the USB.

*Note: Only accessible in device mode.*

Bit 11  **USBSUSP**: USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

*Note: Only accessible in device mode.*

Bit 10  **ESusp**: Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

*Note: Only accessible in device mode.*

Bits 9:8  Reserved, must be kept at reset value.
Bit 7  **GONAKEFF**: Global OUT NAK effective  
Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

*Note: Only accessible in device mode.*

Bit 6  **GINAKEFF**: Global IN non-periodic NAK effective  
Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

*Note: Only accessible in device mode.*

Bit 5  **NPTXFE**: Non-periodic Tx FIFO empty  
This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

*Note: Accessible in host mode only.*

Bit 4  **RXFLVL**: Rx FIFO non-empty  
Indicates that there is at least one packet pending to be read from the Rx FIFO.

*Note: Accessible in both host and device modes.*

Bit 3  **SOF**: Start of frame  
In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt. In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

*Note: This register may return ‘1’ if read immediately after power on reset. If the register bit reads ‘1’ immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.*

*Note: Accessible in both host and device modes.*
Bit 2  **OTGINT**: OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

*Note: Accessible in both host and device modes.*

Bit 1  **MMIS**: Mode mismatch interrupt

The core sets this bit when the application is trying to access:

- A host mode register, when the core is operating in device mode
- A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

*Note: Accessible in both host and device modes.*

Bit 0  **CMOD**: Current mode of operation

Indicates the current mode.

0: Device mode
1: Host mode

*Note: Accessible in both host and device modes.*

### 62.14.7  OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Device mode, see previous section for Host mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

**Address offset**: 0x014

**Reset value**: 0x0040 0020

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**Address offset**: 0x014

**Reset value**: 0x0040 0020

```
<table>
<thead>
<tr>
<th>WKUP</th>
<th>SRQ</th>
<th>DISC</th>
<th>CIDCHG</th>
<th>LPM</th>
<th>PTXFE</th>
<th>HCINT</th>
<th>HPRT</th>
<th>RST</th>
<th>DATAF</th>
<th>SUSP</th>
<th>INCOMP</th>
<th>ISOI</th>
<th>XFR</th>
<th>OEP</th>
<th>EPPINT</th>
<th>Res</th>
<th>Res</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>r</td>
<td>rc_w1</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
| EOPE  | ISOEDR | ENUM  | DNE   | USB   | RST   | USB   | SUSP  | ESUSP | Res   | Res   | GO     | NACK   | EFF   | GI    | NACK  | EFF   | NPTXF | RXF   | LVL   | SOF   | OTG   | INT   | MMIS  | CMOD  |
|-------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| rc_w1 | rc_w1  | rc_w1 | rc_w1 | rc_w1 | r     | r     | r     | r     | r     | r     | rc_w1  | r      | r     | rc_w1 | r     |       |       |       |       |       |       |       |
```
Bit 31 **WKUPINT**: Resume/remote wake-up detected interrupt
Wake-up interrupt during suspend(L2) or LPM(L1) state.
– During suspend(L2):
  In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.
– During LPM(L1):
  This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.
*Note:* Accessible in both device and host modes.

Bit 30 **SRQINT**: Session request/new session detected interrupt
In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when $V_{BUS}$ is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT**: Disconnect detected interrupt
Asserted when a device disconnect is detected.
*Note:* Only accessible in host mode.

Bit 28 **CIDSCHG**: Connector ID status change
The core sets this bit when there is a change in connector ID status.
*Note:* Accessible in both device and host modes.

Bit 27 **LPMINT**: LPM interrupt
In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.
In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG). This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE**: Periodic Tx FIFO empty
Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).
*Note:* Only accessible in host mode.

Bit 25 **HCINT**: Host channels interrupt
The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.
*Note:* Only accessible in host mode.

Bit 24 **HPRTINT**: Host port interrupt
The core sets this bit to indicate a change in port status of one of the OTG_HS controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.
*Note:* Only accessible in host mode.
Bit 23  **RSTDET**: Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

*Note: Only accessible in device mode.*

Bit 22  **DATAFSUSP**: Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake if the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21  **INCOMISOOUT**: Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20  **IISOIXFR**: Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

*Note: Only accessible in device mode.*

Bit 19  **OEPINT**: OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPIINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPIINTx register to clear this bit.

*Note: Only accessible in device mode.*

Bit 18  **IEPINT**: IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPIINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPIINTx register to clear this bit.

*Note: Only accessible in device mode.*

Bits 17:16  Reserved, must be kept at reset value.
Bit 15 **EOPF:** End of periodic frame interrupt
Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.
*Note: Only accessible in device mode.*

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt
The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.
*Note: Only accessible in device mode.*

Bit 13 **ENUMDNE:** Enumeration done
The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.
*Note: Only accessible in device mode.*

Bit 12 **USBRS:** USB reset
The core sets this bit to indicate that a reset is detected on the USB.
*Note: Only accessible in device mode.*

Bit 11 **USBSSP:** USB suspend
The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.
*Note: Only accessible in device mode.*

Bit 10 **ESUSP:** Early suspend
The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.
*Note: Only accessible in device mode.*

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective
Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).
*Note: Only accessible in device mode.*

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective
Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.
*Note: Only accessible in device mode.*

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty
This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).
*Note: Accessible in host mode only.*
Bit 4  **RXFLVL**: Rx FIFO non-empty
Indicates that there is at least one packet pending to be read from the Rx FIFO.
*Note: Accessible in both host and device modes.*

Bit 3  **SOF**: Start of frame
In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.
In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.
*Note: This register may return ‘1’ if read immediately after power on reset. If the register bit reads ‘1’ immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.*
*Note: Accessible in both host and device modes.*

Bit 2  **OTGINT**: OTG interrupt
The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.
*Note: Accessible in both host and device modes.*

Bit 1  **MMIS**: Mode mismatch interrupt
The core sets this bit when the application is trying to access:
– A host mode register, when the core is operating in device mode
– A device mode register, when the core is operating in host mode
The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.
*Note: Accessible in both host and device modes.*

Bit 0  **CMOD**: Current mode of operation
Indicates the current mode.
0: Device mode
1: Host mode
*Note: Accessible in both host and device modes.*
### 62.14.8 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Host mode, see next section for Device mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Mask</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>WUIM</td>
<td>Resume/remote wake-up detected interrupt mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>30</td>
<td>SRQIM</td>
<td>Session request/new session detected interrupt mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>29</td>
<td>DISCINT</td>
<td>Disconnect detected interrupt mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>28</td>
<td>CIDSC</td>
<td>Connector ID status change mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>27</td>
<td>LPMIN</td>
<td>LPM interrupt mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>26</td>
<td>PTXFEM</td>
<td>Periodic Tx FIFO empty mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>25</td>
<td>HCIM</td>
<td>Host channels interrupt mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>24</td>
<td>PRTIM</td>
<td>Host port interrupt mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
<tr>
<td>23</td>
<td>IPXFRM</td>
<td>Incomplete periodic transfer mask</td>
<td>0: Masked interrupt 1: Unmasked interrupt</td>
</tr>
</tbody>
</table>

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 IPXFRM: Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt
Valid for Device mode, see previous section for Host mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>WUIM: Resume/remote wake-up detected interrupt mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1:</td>
<td>Unmasked interrupt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30</th>
<th>SRQIM: Session request/new session detected interrupt mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1:</td>
<td>Unmasked interrupt</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 29</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 28</th>
<th>CIDSCHGM: Connector ID status change mask</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Masked interrupt</td>
</tr>
<tr>
<td>1:</td>
<td>Unmasked interrupt</td>
</tr>
</tbody>
</table>

62.14.9 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Device mode, see previous section for Host mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018
Reset value: 0x0000 0000
Bit 27 **LPMINTM**: LPM interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **RSTDETM**: Reset detected interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 22 **FSUSPM**: Data fetch suspended mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 21 **ISOOXFRM**: Incomplete isochronous OUT transfer mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 20 **ISOIXFRM**: Incomplete isochronous IN transfer mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 19 **OEPINT**: OUT endpoints interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 18 **IEPINT**: IN endpoints interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM**: End of periodic frame interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 14 **ISOODRPM**: Isochronous OUT packet dropped interrupt mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 13 **ENUMDNEM**: Enumeration done mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 12 **USBRST**: USB reset mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 11 **USBSUSPM**: USB suspend mask  
0: Masked interrupt  
1: Unmasked interrupt

Bit 10 **ESUSPM**: Early suspend mask  
0: Masked interrupt  
1: Unmasked interrupt

Bits 9:8 Reserved, must be kept at reset value.
Bit 7 **GONAKEFFM**: Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 6 **GINAKEFFM**: Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **RXFLVLM**: Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM**: Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT**: OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM**: Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

### 62.14.10 OTG receive status debug read register [alternate]
(OTG_GRXSTSR)

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-28 Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27 <strong>STSPHST</strong>: Status phase start</td>
</tr>
<tr>
<td>Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.</td>
</tr>
<tr>
<td>Bits 26-25 Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bits 24:21 **FRMNUM[3:0]:** Frame number  
This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status  
Indicates the status of the received packet  
0001: Global OUT NAK (triggers an interrupt)  
0010: OUT data packet received  
0011: OUT transfer completed (triggers an interrupt)  
0100: SETUP transaction completed (triggers an interrupt)  
0110: SETUP data packet received  
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID  
Indicates the data PID of the received OUT data packet  
00: DATA0  
10: DATA1  
01: DATA2  
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count  
Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number  
Indicates the endpoint number to which the current received packet belongs.

62.14.11 **OTG receive status debug read register [alternate]**  
(OTG_GRXSTSR)

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000
USB on-the-go high-speed (OTG_HS) RM0477

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
- 0010: IN data packet received
- 0011: IN transfer completed (triggers an interrupt)
- 0101: Data toggle error (triggers an interrupt)
- 0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
- 00: DATA0
- 10: DATA1
- 01: DATA2
- 11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

### 62.14.12 OTG status read and pop registers (OTG_GRXSTSP)

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DPID[0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCNT[10:0]</td>
<td>EPNUM[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.
This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020
Reset value: 0x0000 0000

### OTG status read and pop registers [alternate] (OTG_GRXSTSP)

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31-24</th>
<th>Bits 23-21</th>
<th>Bits 20-17</th>
<th>Bits 16-15</th>
<th>Bits 14-11</th>
<th>Bits 10-8</th>
<th>Bits 7-4</th>
<th>Bits 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRMNUM[3:0]</td>
<td>Frame number</td>
<td>This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.</td>
<td>0001: Global OUT NAK (triggers an interrupt)</td>
<td>0010: OUT data packet received</td>
<td>0011: OUT transfer completed (triggers an interrupt)</td>
<td>0100: SETUP transaction completed (triggers an interrupt)</td>
<td>0110: SETUP data packet received</td>
</tr>
<tr>
<td>PKTSTS[3:0]</td>
<td>Packet status</td>
<td>Indicates the status of the received packet</td>
<td>0001: Global OUT NAK (triggers an interrupt)</td>
<td>0010: OUT data packet received</td>
<td>0011: OUT transfer completed (triggers an interrupt)</td>
<td>0100: SETUP transaction completed (triggers an interrupt)</td>
<td>0110: SETUP data packet received</td>
</tr>
<tr>
<td>DPID[1:0]</td>
<td>Data PID</td>
<td>Indicates the data PID of the received OUT data packet</td>
<td>00: DATA0</td>
<td>10: DATA1</td>
<td>01: DATA2</td>
<td>11: MDATA</td>
<td></td>
</tr>
<tr>
<td>BCNT[10:0]</td>
<td>Byte count</td>
<td>Indicates the byte count of the received data packet.</td>
<td>000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EPNUM[3:0]</td>
<td>Endpoint number</td>
<td>Indicates the endpoint number to which the current received packet belongs.</td>
<td>000000000000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Table

<table>
<thead>
<tr>
<th>Bits 31-24</th>
<th>Bits 23-21</th>
<th>Bits 20-17</th>
<th>Bits 16-15</th>
<th>Bits 14-11</th>
<th>Bits 10-8</th>
<th>Bits 7-4</th>
<th>Bits 3-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>PKTSTS[3:0]</td>
<td>DPID</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DPID</td>
<td>BCNT[10:0]</td>
<td>CHNUM[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 PKTSTS[3:0]: Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 DPID[1:0]: Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 BCNT[10:0]: Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 CHNUM[3:0]: Channel number
Indicates the channel number to which the current received packet belongs.

62.14.14 OTG receive FIFO size register (OTG_GRXFSIZ)

The application can program the RAM size that must be allocated to the Rx FIFO.

Address offset: 0x024
Reset value: 0x0000 0400

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 RXFD[15:0]: Rx FIFO depth
This value is in terms of 32-bit words.
Maximum value is 1024
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.
62.14.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)

Valid for Host mode, see next section for Device mode.
Address offset: 0x028
Reset value: 0x0200 0200

<table>
<thead>
<tr>
<th>NPTXFD[15:0]</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 **NPTXFD[15:0]**: Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]**: Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

62.14.16 Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)

Valid for Device mode, see previous section for Host mode.
Address offset: 0x028
Reset value: 0x0200 0200

<table>
<thead>
<tr>
<th>TX0FD[15:0]</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

| TX0FSA[15:0] | rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw |

Bits 31:16 **TX0FD[15:0]**: Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA[15:0]**: Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.
62.14.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Note: In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

Address offset: 0x02C
Reset value: 0x0008 0400

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>NPTXQTOP[6:0]</th>
<th>NPTQXSAV[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPTXFSAV[15:0]</td>
<td>Bits 30:24 NPTXQTOP[6:0]: Top of the non-periodic transmit request queue</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Entry in the non-periodic Tx request queue that is currently being processed by the MAC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bits 30:27: Channel/endpoint number</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bits 26:25: XXXX00X: IN/OUT token</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XXXX01X: Zero-length transmit packet (device IN/host OUT)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XXXX11X: Channel halt command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 24: Terminate (last entry for selected channel/endpoint)</td>
<td></td>
</tr>
</tbody>
</table>

| Bit 23:16 | NPTQXSAV[7:0]: Non-periodic transmit request queue space available |
| | Indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests. |
| | 0: Non-periodic transmit request queue is full |
| | 1: 1 location available |
| | 2: locations available |
| | n: n locations available (0 ≤ n ≤ 8) |
| | Others: Reserved |

| Bit 15:0 | NPTXFSAV[15:0]: Non-periodic Tx FIFO space available |
| | Indicates the amount of free space available in the non-periodic Tx FIFO. Values are in terms of 32-bit words. |
| | 0: Non-periodic Tx FIFO is full |
| | 1: 1 word available |
| | 2: 2 words available |
| | n: n words available (where 0 ≤ n ≤ 512) |
| | Others: Reserved |
**62.14.18 OTG general core configuration register (OTG_GCCFG)**

This register is available in host and device modes.

Address offset: 0x038

Reset value: 0x0000 XXXX

| Bit 31:29 | Reserved, must be kept at reset value. |
| Bit 28   | Reserved, must be kept at reset value. |
| Bit 27   | Reserved, must be kept at reset value. |
| Bit 26   | Reserved, must be kept at reset value. |
| Bit 25   | **FORCEHOSTPD**: Force host mode pull-downs  
            If the ID pin functions are enabled, the host mode pull-downs on DP and DM activate automatically. However, whenever that is not the case, yet host mode is required, this bit must be used to force the pull-downs active.  
            0: Do not force host mode pull-downs  
            1: Force host mode pull-downs |
| Bit 24   | **VBVALOVEN**: Enables a software override of the VBUS B-session detection.  
            0: Use hardware  
            1: Use VBVALOVAL to indicate B-session active |
| Bit 23   | **VBVALOVAL**: Software override value of the VBUS B-session detection  
            0: B-session inactive  
            1: B-session active |
| Bit 22   | **SDEN**: Secondary detection enable  
            0: Secondary detection disabled  
            1: Secondary detection enabled |
| Bit 21   | **VBDEN**: VBUS detection enable  
            Enables VBUS Sensing Comparators in order to detect VBUS presence and/or perform OTG operation.  
            0: VBUS detection disabled  
            1: VBUS detection enabled |
| Bit 20   | **PDEN**: Primary detection enable  
            0: Primary detection disabled  
            1: Primary detection enabled |
| Bit 19   | **DCDEN**: Data Contact Detection enable  
            0: Data Contact Detection disabled  
            1: Data Contact Detection enabled |
Bit 18 **HVDMSRCEN**: Host CDP port Voltage source enable on DM
0: DM voltage source disabled
1: DM Voltage source enabled

Bit 17 **HCDPDETEN**: Host CDP port voltage detector enable on DP
0: DP voltage detection disabled
1: DP voltage detection enabled

Bit 16 **HCDPEN**: Host CDP behavior enable
0: Disable CDP behavior
1: Enable CDP behavior

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **SESSVLD**: VBUS session indicator
Indicates if VBUS is above VBUS session threshold.
0: VBUS is below VBUS session threshold
1: VBUS is above VBUS session threshold

Bit 2 **FSVMINUS**: Single-Ended DM indicator
This bit gives the voltage level on DM (also result of the comparison with $V_{LGC}$ threshold as defined in BC v1.2 standard).
0: DM voltage at low level
1: DM voltage at high level

Bit 1 **FSVPLUS**: Single-Ended DP indicator
This bit gives the voltage level on DP (also result of the comparison with $V_{LGC}$ threshold as defined in BC v1.2 standard).
0: DM voltage at low level
1: DM voltage at high level

Bit 0 **CHGDET**: Charger detection, result of the current mode (primary or secondary).
0: Low value on pin
1: High value on pin

62.14.19 **OTG core ID register (OTG_CID)**

This is a register containing the Product ID as reset value.

Address offset: 0x03C
Reset value: 0x0000 5000

<table>
<thead>
<tr>
<th>Bit 31-16</th>
<th>Bit 15-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 5 1 4 1 3 1 2 1 1 1 0 9876543210</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 31-0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
</tr>
</tbody>
</table>

**PRODUCT_ID[31:0]**: Product ID field
Application-programmable ID field.
### 62.14.20 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x054
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td><strong>ENBESL</strong> Enable best effort service latency</td>
</tr>
<tr>
<td></td>
<td>This bit enables the BESL feature as defined in the LPM errata:</td>
</tr>
<tr>
<td></td>
<td>0: The core works as described in the following document:</td>
</tr>
<tr>
<td></td>
<td>USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0</td>
</tr>
<tr>
<td></td>
<td>specification, July 16, 2007</td>
</tr>
<tr>
<td></td>
<td>1: The core works as described in the LPM Errata:</td>
</tr>
<tr>
<td></td>
<td>Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007</td>
</tr>
<tr>
<td></td>
<td>Note: Only the updated behavior (described in LPM Errata) is considered in</td>
</tr>
<tr>
<td></td>
<td>this document and so the ENBESL bit must be set to ‘1’ by application SW.</td>
</tr>
<tr>
<td>27:25</td>
<td><strong>LPMRCNTSTS[2:0]</strong>: LPM retry count status</td>
</tr>
<tr>
<td></td>
<td>Number of LPM host retries still remaining to be transmitted for the current</td>
</tr>
<tr>
<td></td>
<td>LPM sequence.</td>
</tr>
<tr>
<td></td>
<td>Note: Accessible only in host mode.</td>
</tr>
<tr>
<td>24</td>
<td><strong>SNDLPM</strong>: Send LPM transaction</td>
</tr>
<tr>
<td></td>
<td>When the application software sets this bit, an LPM transaction containing</td>
</tr>
<tr>
<td></td>
<td>two tokens, EXT and LPM is sent. The hardware clears this bit once a valid</td>
</tr>
<tr>
<td></td>
<td>response (STALL, NYET, or ACK) is received from the device or the core has</td>
</tr>
<tr>
<td></td>
<td>finished transmitting the programmed number of LPM retries.</td>
</tr>
<tr>
<td></td>
<td>Note: This bit must be set only when the host is connected to a local port.</td>
</tr>
<tr>
<td></td>
<td>Note: Accessible only in host mode.</td>
</tr>
<tr>
<td>23:21</td>
<td><strong>LPMRCNT[2:0]</strong>: LPM retry count</td>
</tr>
<tr>
<td></td>
<td>When the device gives an ERROR response, this is the number of additional</td>
</tr>
<tr>
<td></td>
<td>LPM retries that the host performs until a valid device response (STALL, NYET,</td>
</tr>
<tr>
<td></td>
<td>or ACK) is received.</td>
</tr>
<tr>
<td></td>
<td>Note: Accessible only in host mode.</td>
</tr>
<tr>
<td>20:17</td>
<td><strong>LPMCHIDX[3:0]</strong>: LPM Channel Index</td>
</tr>
<tr>
<td></td>
<td>The channel number on which the LPM transaction has to be applied while</td>
</tr>
<tr>
<td></td>
<td>sending an LPM transaction to the local device. Based on the LPM channel</td>
</tr>
<tr>
<td></td>
<td>index, the core automatically inserts the device address and endpoint number</td>
</tr>
<tr>
<td></td>
<td>programmed in the corresponding channel into the LPM transaction.</td>
</tr>
<tr>
<td></td>
<td>Note: Accessible only in host mode.</td>
</tr>
</tbody>
</table>

---

### OTG core LPM configuration register (OTG_GLPMCFG)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>Res.</strong></td>
</tr>
<tr>
<td>30</td>
<td><strong>Res.</strong></td>
</tr>
<tr>
<td>29</td>
<td><strong>Res.</strong></td>
</tr>
<tr>
<td>28</td>
<td><strong>ENBESL</strong></td>
</tr>
<tr>
<td>27</td>
<td><strong>LPMRCNTSTS[2:0]</strong></td>
</tr>
<tr>
<td>26</td>
<td><strong>SNDLPM</strong></td>
</tr>
<tr>
<td>25</td>
<td><strong>LPMRCNT[2:0]</strong></td>
</tr>
<tr>
<td>24</td>
<td><strong>LPMCHIDX[3:0]</strong></td>
</tr>
<tr>
<td>23</td>
<td><strong>L1RSM</strong></td>
</tr>
<tr>
<td>22</td>
<td><strong>OK</strong></td>
</tr>
<tr>
<td>21</td>
<td><strong>LPM</strong></td>
</tr>
<tr>
<td>20</td>
<td><strong>EN</strong></td>
</tr>
<tr>
<td>19</td>
<td><strong>LPMACK</strong></td>
</tr>
<tr>
<td>18</td>
<td><strong>BESL</strong></td>
</tr>
<tr>
<td>17</td>
<td><strong>L1SS</strong></td>
</tr>
<tr>
<td>16</td>
<td><strong>REM</strong></td>
</tr>
<tr>
<td>15</td>
<td><strong>BESLTHRS[3:0]</strong></td>
</tr>
<tr>
<td>14</td>
<td><strong>L1DS</strong></td>
</tr>
<tr>
<td>13</td>
<td><strong>LPMRSP[1:0]</strong></td>
</tr>
<tr>
<td>12</td>
<td><strong>SLP</strong></td>
</tr>
<tr>
<td>11</td>
<td><strong>STS</strong></td>
</tr>
</tbody>
</table>

**Bits 31:29** Reserved, must be kept at reset value.

**Bit 28** **ENBESL**: Enable best effort service latency

This bit enables the BESL feature as defined in the LPM errata:

0: The core works as described in the following document:

USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007

1: The core works as described in the LPM Errata:

Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007

Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit must be set to ‘1’ by application SW.

**Bits 27:25** **LPMRCNTSTS[2:0]**: LPM retry count status

Number of LPM host retries still remaining to be transmitted for the current LPM sequence.

Note: Accessible only in host mode.

**Bit 24** **SNDLPM**: Send LPM transaction

When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.

Note: This bit must be set only when the host is connected to a local port.

Note: Accessible only in host mode.

**Bits 23:21** **LPMRCNT[2:0]**: LPM retry count

When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.

Note: Accessible only in host mode.

**Bits 20:17** **LPMCHIDX[3:0]**: LPM Channel Index

The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.

Note: Accessible only in host mode.
Bit 16  **L1RSMOK**: Sleep state resume OK
Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 μs ($T_{L1\text{Residency}}$).
This bit is reset when SLPSTS is 0.
0: The application or host cannot start resume from Sleep state
1: The application or host can start resume from Sleep state

Bit 15  **SLPSTS**: Port sleep status
Device mode:
This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1\text{TokenRetry}}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.
The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.
The core comes out of sleep:
– When there is any activity on the USB linestate
– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.
Host mode:
The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.
The core clears this bit after:
– The core detects a remote L1 wake-up signal,
– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or
– The application sets the L1Resume/ remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).
0: Core not in L1
1: Core in L1

Bits 14:13  **LPMRSP[1:0]**: LPM response
Device mode:
The response of the core to LPM transaction received is reflected in these two bits.
Host mode:
Handshake response received from local device for LPM transaction
11: ACK
10: NYET
01: STALL
00: ERROR (No handshake response)

Bit 12  **L1DSEN**: L1 deep sleep enable
Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to ‘1’ by application SW in all the cases.
Bits 11:8 **BESLTHRS[3:0]**: BESL threshold

**Device mode:**
The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

**Host mode:**
The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host (T_{L1HubDrvResume2}) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum T_{L1HubDrvResume2}.

**Thres[3:0]** host mode resume signaling time (μs):
- 0000: 75
- 0001: 100
- 0010: 150
- 0011: 250
- 0100: 350
- 0101: 450
- 0110: 950
- All other values: reserved

Bit 7 **L1SSEN**: L1 Shallow Sleep enable
Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to ‘1’ by application SW in all the cases.

Bit 6 **REMWAKE**: bRemoteWake value

**Host mode:**
The value of remote wake up to be sent in the wIndex field of LPM transaction.

**Device mode (read-only):**
This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.
Bits 5:2 **BESL[3:0]**: Best effort service latency

**Host mode:**
The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{LHubDrvResume}$ for host initiated resume.

**Device mode (read-only):**
This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

$BESL[3:0]T_{BESL}$ (μs)

<table>
<thead>
<tr>
<th>Value</th>
<th>BESL Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>125</td>
</tr>
<tr>
<td>0001</td>
<td>150</td>
</tr>
<tr>
<td>0010</td>
<td>200</td>
</tr>
<tr>
<td>0011</td>
<td>300</td>
</tr>
<tr>
<td>0100</td>
<td>400</td>
</tr>
<tr>
<td>0101</td>
<td>500</td>
</tr>
<tr>
<td>0110</td>
<td>1000</td>
</tr>
<tr>
<td>0111</td>
<td>2000</td>
</tr>
<tr>
<td>1000</td>
<td>3000</td>
</tr>
<tr>
<td>1001</td>
<td>4000</td>
</tr>
<tr>
<td>1010</td>
<td>5000</td>
</tr>
<tr>
<td>1011</td>
<td>6000</td>
</tr>
<tr>
<td>1100</td>
<td>7000</td>
</tr>
<tr>
<td>1101</td>
<td>8000</td>
</tr>
<tr>
<td>1110</td>
<td>9000</td>
</tr>
<tr>
<td>1111</td>
<td>10000</td>
</tr>
</tbody>
</table>

Bit 1 **LPMACK**: LPM token acknowledge enable
Handshake response to LPM token preprogrammed by device application software.

1: ACK
Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:
- No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
- Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
- No data pending in transmit queue (else NYET).

0: NYET
The preprogrammed software bit is over-ridden for response to LPM token when:
- The received bLinkState is not L1 (STALL response), or
- An error is detected in either of the LPM token packets because of corruption (ERROR response).

**Note:** Accessible only in device mode.

Bit 0 **LPMEN**: LPM support enable
The application uses this bit to control the OTG_HS core LPM capabilities.
If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.
If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled
1: LPM capability is enabled
62.14.21 OTG host periodic transmit FIFO size register  
(OTG_HPTXFSIZ)

Address offset: 0x100
Reset value: 0x0400 0800

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>PTXFSIZ[15:0]</th>
<th>Host periodic Tx FIFO depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This value is in terms of 32-bit words.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum value is 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>PTXSA[15:0]</th>
<th>Host periodic Tx FIFO start address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This field configures the memory start address for periodic transmit FIFO RAM.</td>
</tr>
</tbody>
</table>

62.14.22 OTG device IN endpoint transmit FIFO x size register  
(OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 8)
Reset value: 0x0200 0400, 0x0200 0600, 0x0200 0800, 0x0200 0A00, 0x0200 0C00, 0x0200 0E00, 0x0200 1000, 0x0200 1200

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>INEPTXFD[15:0]</th>
<th>IN endpoint Tx FIFO depth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This value is in terms of 32-bit words.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Minimum value is 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15:0</th>
<th>INEPTXSA[15:0]</th>
<th>IN endpoint FIFOx transmit RAM start address</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.</td>
</tr>
</tbody>
</table>

62.14.23 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified. Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:
62.14.24 OTG host configuration register (OTG_HCFG)

This register configures the core after power-on. Do not make changes to this register after initializing the host.

Address offset: 0x400
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

FSLSS    FSLSPCS[1:0]

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

Bit 2 FSLSS: FS- and LS-only support

The application uses this bit to control the core enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

Bits 1:0 FSLSPCS[1:0]: FS/LS PHY clock select

<table>
<thead>
<tr>
<th>Condition</th>
<th>FS host mode</th>
<th>LS host mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>01:</td>
<td>PHY clock is running at 48 MHz</td>
<td>Reserved</td>
</tr>
<tr>
<td>Others:</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

62.14.25 OTG host frame interval register (OTG_HFIR)

This register stores the frame interval information for the current speed to which the OTG_HS controller has enumerated.

Address offset: 0x404
Reset value: 0x0000 EA60

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

FRIVL[15:0]

rw    rw    rw    rw    rw    rw    rw    rw    rw    rw    rw    rw    rw    rw    rw    rw

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).
Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RLDCTRL**: Reload control
This bit allows dynamic reloading of the HFIR register during run time.
- 0: The HFIR cannot be reloaded dynamically
- 1: The HFIR can be dynamically reloaded during run time.
This bit needs to be programmed during initial configuration and its value must not be changed during run time.

**Caution**: RLDCTRL = 0 is not recommended.

Bits 15:0 **FRIVL[15:0]**: Frame interval
The value that the application programs to this field, specifies the interval between two consecutive micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

- Frame interval = 125 μs × (FRIVL - 1) in high speed operation
- Frame interval = 1 ms × (FRIVL - 1) in low/full speed operation

### 62.14.26 OTG host frame number/frame time remaining register (OTG_HFNUM)
This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

Address offset: 0x408
Reset value: 0x0000 3FFF

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td></td>
</tr>
</tbody>
</table>

**FTREM[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td></td>
</tr>
</tbody>
</table>

**FRNUM[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td>_</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 **FTREM[15:0]**: Frame time remaining
Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]**: Frame number
This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.
62.14.27  OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

Address offset: 0x410
Reset value: 0x0008 0100

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTXQTOP[7:0]</td>
<td>PTXQSAV[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

PTXFSAVL[15:0]

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

Bits 31:24  PTXQTOP[7:0]: Top of the periodic transmit request queue
This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.
This register is used for debugging.
Bit 31: Odd/Even frame
0XXXXXXX: send in even frame
1XXXXXXX: send in odd frame
Bits 30:27: Channel/endpoint number
Bits 26:25: Type
XXXXX00X: IN/OUT
XXXXX01X: Zero-length packet
XXXXX11X: Disable channel command
Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16  PTXQSAV[7:0]: Periodic transmit request queue space available
Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.
0: Periodic transmit request queue is full
1: 1 location available
2: 2 locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0  PTXFSAVL[15:0]: Periodic transmit data FIFO space available
Indicates the number of free locations available to be written to in the periodic Tx FIFO.
Values are in terms of 32-bit words
0: Periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ PTXFD)
Others: Reserved
When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in Figure 921. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

Address offset: 0x414
Reset value: 0x0000 0000

Bits 31:16  Reserved, must be kept at reset value.
Bits 15:0  HAIN[T[15:0]: Channel interrupts
  One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

Address offset: 0x418
Reset value: 0x0000 0000

Bits 31:16  Reserved, must be kept at reset value.
Bits 15:0  HAIN[TM[15:0]: Channel interrupt mask
  0: Masked interrupt
  1: Unmasked interrupt
  One bit per channel: Bit 0 for channel 0, bit 15 for channel 15
62.14.30 OTG host port control and status register (OTG_HPRT)

This register is available only in host mode. Currently, the OTG host supports only one port. A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in Figure 921. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

Address offset: 0x440
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Res.</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 18  | PSPD[1:0]     | Port speed
Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved
00: High speed
| 17  | PTCTL[3:0]    | Port test control
The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved
| 16  | PPWR          | Port power
The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition. Note that this bit does not directly activate the voltage on the connector VBUS.
0: Power off
1: Power on
| 15  |              |                                            |
| 14  |              |                                            |
| 13  |              |                                            |
| 12  |              |                                            |
| 11  | PLSTS[1:0]    | Port line status
Indicates the current logic level USB data lines
Bit 10: Logic level of OTG_HS_DP
Bit 11: Logic level of OTG_HS_DM
| 10  |              |                                            |
| 9   |              |                                            |
| 8   |              |                                            |
| 7   |              |                                            |
| 6   |              |                                            |
| 5   |              |                                            |
| 4   |              |                                            |
| 3   |              |                                            |
| 2   |              |                                            |
| 1   |              |                                            |
| 0   |              |                                            |
Bit 9  Reserved, must be kept at reset value.

Bit 8  **PRST**: Port reset
When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.
0: Port not in reset
1: Port in reset
The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.
High speed: 50 ms
Full speed/Low speed: 10 ms

Bit 7  **PSUSP**: Port suspend
The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.
The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wake-up signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).
0: Port not in suspend mode
1: Port in suspend mode

Bit 6  **PRES**: Port resume
The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.
If the core detects a USB remote wake-up sequence, as indicated by the port resume/remote wake-up detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.
0: No resume driven
1: Resume driven
When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:
1. The application sets this bit to drive resume signaling on the port.
2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.
3. If the core detects a USB remote wake-up sequence, as indicated by the port L1Resume/Remote L1wake-up detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume. This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5  **POCCHNG**: Port overcurrent change
The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4  **POCA**: Port overcurrent active
Indicates the overcurrent condition of the port.
0: No overcurrent condition
1: Overcurrent condition
Bit 3 **PENCHNG:** Port enable/disable change
The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable
A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.
0: Port disabled
1: Port enabled

Bit 1 **PCDET:** Port connect detected
The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status
0: No device is attached to the port
1: A device is attached to the port

62.14.31 **OTG host channel x characteristics register (OTG_HCCHARx)**
Address offset: 0x500 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28:22</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHENA</td>
<td>CHDIS</td>
<td>ODDFRM</td>
<td>DAD[6:0]</td>
</tr>
<tr>
<td>rs</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>EPDIR</td>
<td>EPNUM[3:0]</td>
<td>MPSIZ[10:0]</td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bit 31 CHENA:** Channel enable
This field is set by the application and cleared by the OTG host.
0: Channel disabled
1: Channel enabled

**Bit 30 CHDIS:** Channel disable
The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

**Bit 29 ODDFRM:** Odd frame
This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.
0: Even frame
1: Odd frame

**Bits 28:22 DAD[6:0]:** Device address
This field selects the specific device serving as the data source or sink.
Bits 21:20 **MCNT[1:0]: Multicount**
This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used.
- 00: Reserved. This field yields undefined results
- 01: 1 transaction
- 10: 2 transactions per frame to be issued for this endpoint
- 11: 3 transactions per frame to be issued for this endpoint

*Note: This field must be set to at least 01.*

Bits 19:18 **EPTYP[1:0]: Endpoint type**
Indicates the transfer type selected.
- 00: Control
- 01: Isochronous
- 10: Bulk
- 11: Interrupt

Bit 17 **LSDEV: Low-speed device**
This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR: Endpoint direction**
Indicates whether the transaction is IN or OUT.
- 0: OUT
- 1: IN

Bits 14:11 **EPNUM[3:0]: Endpoint number**
Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]: Maximum packet size**
Indicates the maximum packet size of the associated endpoint.

### 62.14.32 OTG host channel x split control register (OTG_HCSPLTx)
Address offset: 0x504 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XACTPOS[1:0]</th>
<th>HUBADDR[8:0]</th>
<th>PRTADDR[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **SPLITEN**: Split enable
The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT**: Do complete split
The application sets this bit to request the OTG host to perform a complete split transaction.
Bits 15:14  **XACTPOS[1:0]**: Transaction position
This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.
11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)
10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)
00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)
01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7  **HUBADDR[6:0]**: Hub address
This field holds the device address of the transaction translator hub.

Bits 6:0  **PRTADDR[6:0]**: Port address
This field is the port number of the recipient transaction translator.

### 62.14.33 OTG host channel x interrupt register (OTG_HCINTx)
This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in Figure 921. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

Address offset: 0x508 + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
<td>rC_w1</td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

- **Bit 10  DTErr**: Data toggle error.
- **Bit 9  FRMOR**: Frame overrun.
- **Bit 8  BBErr**: Babble error.
- **Bit 7  TXErr**: Transaction error.
  Indicates one of the following errors occurred on the USB.
  - CRC check failure
  - Timeout
  - Bit stuff error
  - False EOP

- **Bit 6  NYET**: Not yet ready response received interrupt.
- **Bit 5  ACK**: ACK response received/transmitted interrupt.
- **Bit 4  NAK**: NAK response received interrupt.
Bit 3  **STALL**: STALL response received interrupt.

Bit 2  **AHBERR**: AHB error
This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1  **CHH**: Channel halted
This indicates that the transfer completed:
- because of any USB transaction error
- in response to disable request by the application
- normally

Bit 0  **XFRC**: Transfer completed.
Transfer completed normally without any errors.

### 62.14.34 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

This register reflects the mask for each channel status described in the previous section.

Address offset: 0x50C + 0x20 * x, (x = 0 to 15)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11  Reserved, must be kept at reset value.

Bit 10  **DTERRM**: Data toggle error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 9  **FRMORM**: Frame overrun mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 8  **BBERRM**: Babble error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 7  **TXERRM**: Transaction error mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 6  **NYET**: response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt
Bit 5 **ACKM**: ACK response received/transmitted interrupt mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 4 **NAKM**: NAK response received interrupt mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 3 **STALLM**: STALL response received interrupt mask.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 2 **AHBERRM**: AHB error.
   0: Masked interrupt
   1: Unmasked interrupt

Bit 1 **CHHM**: Channel halted mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed mask
   0: Masked interrupt
   1: Unmasked interrupt

### 62.14.35 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>DO PNG</th>
<th>DPID[1:0]</th>
<th>PKTCNT[8:0]</th>
<th>XFRSIZ[18:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>31</td>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>29</td>
<td>11</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>28</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>27</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**XFRSIZ[15:0]**

| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

Bit 31 **DOPNG**: Do Ping

This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.

*Note*: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **DPID[1:0]**: Data PID

The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.

00: DATA0
01: DATA2
10: DATA1
11: SETUP (control) / MDATA (non-control)
Bits 28:19  **PKTCNT[9:0]:** Packet count
This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).
The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0  **XFRSIZ[18:0]:** Transfer size
For an OUT, this field is the number of data bytes the host sends during the transfer.
For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

### 62.14.36  OTG host channel x DMA address register (OTG_HCDMAx)

Address offset: $0x514 + 0x20 \times x$, $(x = 0 \text{ to } 15)$
Reset value: $0x0000\ 0000$

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **DMAADDR[31:0]:** DMA address
This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

### 62.14.37  Device-mode registers

These registers must be programmed every time the core changes to device mode

### 62.14.38  OTG device configuration register (OTG_DCFG)

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Address offset: $0x800$
Reset value: $0x0220\ 0000$

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:26  Reserved, must be kept at reset value.
Bits 25:24 **PERSCHIVL[1:0]:** Periodic schedule interval

This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro) frame.

- When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data
- When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field
- After the specified time within a (micro) frame, the DMA switches to fetching nonperiodic endpoints

00: 25% of (micro)frame  
01: 50% of (micro)frame  
10: 75% of (micro)frame  
11: Reserved

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **ERRATIM:** Erratic error interrupt mask

1: Mask early suspend interrupt on erratic error  
0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval

Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.

00: 80% of the frame interval  
01: 85% of the frame interval  
10: 90% of the frame interval  
11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address

The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake

The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer’s status stage.

1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.  
0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.
Bits 1:0 **DSPD[1:0]: Device speed**

Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.

- 00: High speed
- 01: Full speed
- 10: Reserved
- 11: Reserved

### 62.14.39 OTG device control register (OTG_DCTL)

**Address offset:** 0x804

**Reset value:** 0x0000 0002

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>r</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:19 Reserved, must be kept at reset value.

- **Bit 18 DSBESLRJCT:** Deep sleep BESL reject
  Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

- **Bit 11 POPRGDNE:** Power-on programming done
  The application uses this bit to indicate that register programming is completed after a wake-up from power down mode.

- **Bit 10 CGONAK:** Clear global OUT NAK
  Writing 1 to this field clears the Global OUT NAK.

- **Bit 9 SGNOK:** Set global OUT NAK
  Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

- **Bit 8 CGINAK:** Clear global IN NAK
  Writing 1 to this field clears the Global IN NAK.

- **Bit 7 SGINAK:** Set global IN NAK
  Writing 1 to this field sets the Global non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.
Table 674 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

<table>
<thead>
<tr>
<th>Operating speed</th>
<th>Device state</th>
<th>Minimum duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full speed</td>
<td>Suspended</td>
<td>1 ms + 2.5 µs</td>
</tr>
<tr>
<td>Full speed</td>
<td>Idle</td>
<td>2.5 µs</td>
</tr>
<tr>
<td>Full speed</td>
<td>Not Idle or suspended (Performing transactions)</td>
<td>2.5 µs</td>
</tr>
<tr>
<td>High speed</td>
<td>Not Idle or suspended (Performing transactions)</td>
<td>125 µs</td>
</tr>
</tbody>
</table>
### 62.14.40 OTG device status register (OTG_DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

- **Address offset:** 0x808
- **Reset value:** 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEVLNSTS[1:0]</td>
<td>FNSOF[13:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 31:24** Reserved, must be kept at reset value.
- **Bits 23:22** **DEVLNSTS[1:0]:** Device line status
  - Indicating the current logic level USB data lines.
  - Bit [23]: Logic level of D+
  - Bit [22]: Logic level of D-
- **Bits 21:8** **FNSOF[13:0]:** Frame number of the received SOF
  - Bits 7:4 Reserved, must be kept at reset value.
- **Bit 3** **EERR:** Erratic error
  - The core sets this bit to report any erratic errors.
  - Due to erratic errors, the OTG_HS controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.
- **Bits 2:1** **ENUMSPD[1:0]:** Enumerated speed
  - Indicates the speed at which the OTG_HS controller has come up after speed detection through a chirp sequence.
  - **00:** High Speed
  - **01:** Full Speed
  - **11:** Reserved
  - Others: reserved
- **Bit 0** **SUSPSTS:** Suspend status
  - In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:
    - When there is an activity on the USB data lines
    - When the application writes to the remote wake-up signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).
62.14.41 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x810

Reset value: 0x0000 0000

| Bit 31:14 | Reserved, must be kept at reset value. |
| Bit 13 | NAKM: NAK interrupt mask |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
| Bit 12:10 | Reserved, must be kept at reset value. |
| Bit 9 | Reserved, must be kept at reset value. |
| Bit 8 | TXFURM: FIFO underrun mask |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
| Bit 7 | Reserved, must be kept at reset value. |
| Bit 6 | INEPNEM: IN endpoint NAK effective mask |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
| Bit 5 | INEPNMM: IN token received with EP mismatch mask |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
| Bit 4 | ITTXFEMSK: IN token received when Tx FIFO empty mask |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
| Bit 3 | TOM: Timeout condition mask (Non-isochronous endpoints) |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
| Bit 2 | AHBERRM: AHB error mask |
| 0 | Masked interrupt |
| 1 | Unmasked interrupt |
Bit 1 **EPDM**: Endpoint disabled interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

### 62.14.42 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x814

Reset value: 0x0000 0000

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK**: NYET interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 13 **NAKMSK**: NAK interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
   0: Masked interrupt
   1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM**: Out packet error mask
   0: Masked interrupt
   1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUPM**: Back-to-back SETUP packets received mask
   Applies to control OUT endpoints only.
   0: Masked interrupt
   1: Unmasked interrupt
Bit 5 **STSPHRSRM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM**: OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM**: SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM**: AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM**: Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM**: Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

**62.14.43 OTG device all endpoints interrupt register (OTG_DAINT)**

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

Address offset: 0x818
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**OEPINT[15:0]**: OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

**IEPINT[15:0]**: IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.
62.14.44 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

Address offset: 0x81C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16</th>
<th>OEPM[15:0]: OUT EP interrupt mask bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>One per OUT endpoint:</td>
<td></td>
</tr>
<tr>
<td>Bit 16 for OUT EP 0, bit 19 for OUT EP 3</td>
<td></td>
</tr>
<tr>
<td>0: Masked interrupt</td>
<td></td>
</tr>
<tr>
<td>1: Unmasked interrupt</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0</th>
<th>IEPM[15:0]: IN EP interrupt mask bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>One bit per IN endpoint:</td>
<td></td>
</tr>
<tr>
<td>Bit 0 for IN EP 0, bit 3 for IN EP 3</td>
<td></td>
</tr>
<tr>
<td>0: Masked interrupt</td>
<td></td>
</tr>
<tr>
<td>1: Unmasked interrupt</td>
<td></td>
</tr>
</tbody>
</table>

62.14.45 OTG device threshold control register (OTG_DTHRCTL)

Address offset: 0x0830
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td>ARPEN: Arbiter parking enable</td>
</tr>
<tr>
<td>This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.</td>
<td></td>
</tr>
<tr>
<td>Bit 26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bits 25:17 **RXTHRLEN[8:0]:** Receive threshold length  
This field specifies the receive thresholding size in 32-bit words. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight 32-bit words. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 16 **RXTHREN:** Receive threshold enable  
When this bit is set, the core enables thresholding in the receive direction.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:2 **TXTHRLEN[8:0]:** Transmit threshold length  
This field specifies the transmit thresholding size in 32-bit words. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO before the core can start transmitting on the USB. The threshold length has to be at least eight 32-bit words. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 1 **ISOTHREN:** ISO IN endpoint threshold enable  
When this bit is set, the core enables thresholding for isochronous IN endpoints.

Bit 0 **NONISOTHREN:** Nonisochronous IN endpoints threshold enable  
When this bit is set, the core enables thresholding for nonisochronous IN endpoints.

### 62.14.46 OTG device IN endpoint FIFO empty interrupt mask register  
(OTG_DIEPEMMPMSK)

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

Address offset: 0x834

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**INEPTXFEM[15:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits  
These bits act as mask bits for OTG_DIEPINTx.

TXFE interrupt one bit per IN endpoint:

- Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
- 0: Masked interrupt
- 1: Unmasked interrupt
### 62.14.47 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>EPENA</th>
<th>EPDIS</th>
<th>SD1 PID</th>
<th>SD0 PID</th>
<th>SNAK</th>
<th>CNAK</th>
<th>TXFNUM[3:0]</th>
<th>STALL</th>
<th>Res.</th>
<th>EPTYP[1:0]</th>
<th>NAK STS</th>
<th>DPI0</th>
<th>DPI1</th>
</tr>
</thead>
<tbody>
<tr>
<td>rs</td>
<td>rs</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Bit 31 EPENA: Endpoint enable
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

#### Bit 30 EPDIS: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

#### Bit 29 SD1PID: Set DATA1 PID
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

#### Bit 28 SD0PID: Set DATA0 PID
Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

#### Bit 27 SNAK: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

#### Bit 26 CNAK: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

#### Bits 25:22 TXFNUM[3:0]: Tx FIFO number
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.
Bit 21  **STALL**: STALL handshake  
Applies to non-control, non-isochronous IN endpoints only (access type is rw).  
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK  
bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.  
Only the application can clear this bit, never the core.  

Bit 20  Reserved, must be kept at reset value.  

Bits 19:18  **EPTYP[1:0]**: Endpoint type  
This is the transfer type supported by this logical endpoint.  
00: Control  
01: Isochronous  
10: Bulk  
11: Interrupt  

Bit 17  **NAKSTS**: NAK status  
It indicates the following:  
0: The core is transmitting non-NAK handshakes based on the FIFO status.  
1: The core is transmitting NAK handshakes on this endpoint.  
When either the application or the core sets this bit:  
For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint,  
even if there are data available in the Tx FIFO.  
For isochronous IN endpoints: The core sends out a zero-length data packet, even if there  
are data available in the Tx FIFO.  
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an  
ACK handshake.  

Bit 16  **DPID**: Endpoint data PID  
Applies to interrupt/bulk IN endpoints only.  
Contains the PID of the packet to be received or transmitted on this endpoint. The  
application must program the PID of the first packet to be received or transmitted on this  
endpoint, after the endpoint is activated. The application uses the SD0PID and SD1PID  
register fields to program either DATA0 or DATA1 PID.  
0: DATA0  
1: DATA1  

Bit 15  **USBAEP**: USB active endpoint  
Indicates whether this endpoint is active in the current configuration and interface. The core  
clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving  
the SetConfiguration and SetInterface commands, the application must program endpoint  
registers accordingly and set this bit.  

Bits 14:11  Reserved, must be kept at reset value.  

Bits 10:0  **MPSIZ[10:0]**: Maximum packet size  
The application must program this field with the maximum packet size for the current logical  
endpoint. This value is in bytes.
62.14.48 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPENA</td>
<td>EPDIS</td>
<td>SODDFRM</td>
<td>SEVNFRM</td>
<td>SNAK</td>
<td>CNAK</td>
<td>TXFNUM[3:0]</td>
<td>STALL</td>
<td>Res.</td>
<td>EPTYP[1:0]</td>
<td>NAKSTS</td>
<td>EO NUM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rs</td>
<td>rs</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 EPENA: Endpoint enable
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 EPDIS: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before
the transfer for that endpoint is complete. The application must wait for the endpoint
disabled interrupt before treating the endpoint as disabled. The core clears this bit before
setting the endpoint disabled interrupt. The application must set this bit only if endpoint
enable is already set for this endpoint.

Bit 29 SODDFRM: Set odd frame
Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 SEVNFRM: Set even frame
Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 SNAK: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an
endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt,
or after a SETUP is received on the endpoint.

Bit 26 CNAK: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 TXFNUM[3:0]: Tx FIFO number
These bits specify the FIFO number associated with this endpoint. Each active IN endpoint
must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.
Bit 21 **STALL**: STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]**: Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS**: NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM**: Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame
1: Odd frame

Bit 15 **USBAEP**: USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.
62.14.49  OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in Figure 921. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0x908 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>NAK: NAK input</td>
</tr>
<tr>
<td>26</td>
<td>The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>PKTDRPSTS: Packet dropped status</td>
</tr>
<tr>
<td>23</td>
<td>This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>TXFIFOUDRN: Transmit Fifo Underrun (TxfifoUndrm)</td>
</tr>
<tr>
<td>19</td>
<td>The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled</td>
</tr>
<tr>
<td>18</td>
<td>TXFE: Transmit FIFO empty</td>
</tr>
<tr>
<td>17</td>
<td>This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).</td>
</tr>
<tr>
<td>16</td>
<td>INEPNE: IN endpoint NAK effective</td>
</tr>
<tr>
<td>15</td>
<td>This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.</td>
</tr>
</tbody>
</table>
Bit 5 **INEPNM**: IN token received with EP mismatch
 Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE**: IN token received when Tx FIFO is empty
 Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC**: Timeout condition
 Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 **AHBERR**: AHB error
 This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD**: Endpoint disabled interrupt
 This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC**: Transfer completed interrupt
 This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

**62.14.50 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)**

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

Address offset: 0x910

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
<th>PKTCNT[1:0]</th>
<th>XFRSIZ[6:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]**: Packet count
 Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.
 This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.
62.14.51 OTG device IN endpoint x DMA address register
(OTG_DIEPDMAX)

Address offset: 0x914 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 DMAADDR[31:0]: DMA Address
This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

62.14.52 OTG device IN endpoint transmit FIFO status register
(OTG_DTXFSTSx)

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

Address offset: 0x918 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0200

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INEPTFSAV[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 INEPTFSAV[15:0]: IN endpoint Tx FIFO space available
Indicates the amount of free space available in the endpoint Tx FIFO.
Values are in terms of 32-bit words:
0x0: Endpoint Tx FIFO is full
0x1: 1 word available
0x2: 2 words available
0xn: n words available
Others: Reserved
62.14.53 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0x910 + 0x20 * x, (x = 1 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 MCNT[1:0]: Multi count
For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 PKTCNT[9:0]: Packet count
Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 XFRSIZ[18:0]: Transfer size
This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

62.14.54 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–8.

Address offset: 0xB00
Reset value: 0x0000 8000
Bit 31 EPENA: Endpoint enable
The application sets this bit to start transmitting data on endpoint 0. The core clears this bit before setting any of the following interrupts on this endpoint:
- SETUP phase done
- Endpoint disabled
- Transfer completed

Bit 30 EPDIS: Endpoint disable
The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 SNAK: Set NAK
A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 CNAK: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 STALL: STALL handshake
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 SNPM: Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 EPTYP[1:0]: Endpoint type
Hardcoded to 00 for control.

Bit 17 NAKSTS: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 USBAEP: USB active endpoint
This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.
62.14.55 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in Figure 921. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0xB08 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0080

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **STPKTRX:** Setup packet received
Applicable for control OUT endpoints in only in the Buffer DMA Mode. Set by the OTG_HS, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per buffer. On receiving a setup packet, the OTG_HS closes the buffer and disables the corresponding endpoint after SETUP_COMPLETE status is seen in the Rx FIFO. OTG_HS puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN or OUT token after the SETUP packet for that particular endpoint. The application must then re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Because of the above behavior, OTG_HS can receive any number of back to back setup packets and one buffer for every setup packet is used.

Bit 14 **NYET:** NYET interrupt
This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint.

Bit 13 **NAK:** NAK input
The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.
Bit 12 **BERR**: Babble error interrupt
The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERR**: OUT packet error
This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. This interrupt is valid only when thresholding is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP**: Back-to-back SETUP packets received
Applies to control OUT endpoint only.
This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 **STSPHSRX**: Status phase received for control write
This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG_HS has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS**: OUT token received when endpoint disabled
Applies only to control OUT endpoints.
Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP**: SETUP phase done
Applies to control OUT endpoint only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 **_AHBERR**: AHB error
This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD**: Endpoint disabled interrupt
This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC**: Transfer completed interrupt
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.
62.14.56  **OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)**

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–8.

Address offset: 0xB10

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30–29</th>
<th>Bit 28–20</th>
<th>Bit 19</th>
<th>Bit 18–7</th>
<th>Bit 6–0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res.</td>
<td>STUPCNT[1:0]</td>
<td>Res.</td>
<td>PKTCNT</td>
<td>Res.</td>
<td>XFRSIZ[6:0]</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30:29 | STUPCNT[1:0]: SETUP packet count |
| This field specifies the number of back-to-back SETUP data packets the endpoint can receive. |
| 01: 1 packet |
| 10: 2 packets |
| 11: 3 packets |

| Bit 28:20 | Reserved, must be kept at reset value. |

| Bit 19 | PKTCNT: Packet count |
| This field is decremented to zero after a packet is written into the Rx FIFO. |

| Bit 18:7 | Reserved, must be kept at reset value. |

| Bit 6:0 | XFRSIZ[6:0]: Transfer size |
| Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. |
| The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory. |
62.14.57 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)

Address offset: 0xB14 + 0x20 * x, (x = 0 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:16 DMAADDR[31:16]</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0 DMAADDR[15:0]</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
</table>

Bits 31:0 DMAADDR[31:0]: DMA Address
This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

62.14.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0 USBA</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 15:0 EPENA</th>
<th>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 23:15 SD1 PID</th>
<th>rs rs w w w w w w w w w w w w w w</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 22:15 SD0 PID</td>
<td>rs rs w w w w w w w w w w w w w w</td>
</tr>
</tbody>
</table>

Bit 31 *EPENA*: Endpoint enable
Applies to IN and OUT endpoints.
The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 *EPDIS*: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.
Bit 29 **SD1PID**: Set DATA1 PID
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID**: Set DATA0 PID
Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK**: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL**: STALL handshake
Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM**: Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]**: Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS**: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.
Bit 16 **DPID**: Endpoint data PID  
Applies to interrupt/bulk OUT endpoints only.  
Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID and SD1PID register fields to program either DATA0 or DATA1 PID.  
0: DATA0  
1: DATA1

Bit 15 **USBAEP**: USB active endpoint  
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size  
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

### 62.14.59 OTG device OUT endpoint x control register [alternate]  
(OTG_DOEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

| Bit 31 | EPENA: Endpoint enable | Applies to IN and OUT endpoints.  
The application sets this bit to start transmitting data on an endpoint.  
The core clears this bit before setting any of the following interrupts on this endpoint:  
– SETUP phase done  
– Endpoint disabled  
– Transfer completed |
Bit 30 **EPDIS**: Endpoint disable
The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM**: Set odd frame
Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNF RM**: Set even frame
Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK**: Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK**: Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL**: STALL handshake
Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM**: Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]**: Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS**: NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit’s setting, the core always responds to SETUP data packets with an ACK handshake.
Bit 16 **EONUM**: Even/odd frame
Applies to isochronous OUT endpoints only.
Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.
0: Even frame
1: Odd frame

Bit 15 **USBAEP**: USB active endpoint
Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]**: Maximum packet size
The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

### 62.14.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0xB10 + 0x20 * x, (x = 1 to 8)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**XFRSIZ[15:0]**

Bit 31: Reserved, must be kept at reset value.
USB on-the-go high-speed (OTG_HS)  

62.14.61 OTG power and clock gating control register (OTG_PCGCCTL)

This register is available in host and device modes.

Address offset: 0xE00

Reset value: 0x200B 8000

Bits 30:29 **RXDPID[1:0]:**
- **Condition:** isochronous OUT endpoints
- Received data PID
- This is the data PID received in the last packet for this endpoint.
- 00: DATA0
- 01: DATA2
- 10: DATA1
- 11: MDATA

**Condition:** control OUT endpoints

STUPCNT[1:0]: SETUP packet count
- This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
- 01: 1 packet
- 10: 2 packets
- 11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count
- Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
- This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size
- This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
- The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

This register is available in host and device modes.

Address offset: 0xE00

Reset value: 0x200B 8000

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Reserved</th>
<th>Must be kept at reset value.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bit 7</th>
<th><strong>SUSP:</strong> Deep Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the PHY is in Deep Sleep when in L1 state.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 6</th>
<th><strong>PHYSLEEP:</strong> PHY in Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This bit indicates that the PHY is in the Sleep state.</td>
</tr>
</tbody>
</table>
### Bit 5 **ENL1GTG**: Enable sleep clock gating
When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_i1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

### Bit 4 **PHYSUSP**: PHY suspended
Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

### Bits 3:2 Reserved, must be kept at reset value.

### Bit 1 **GATEHCLK**: Gate HCLK
The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wake-up logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

### Bit 0 **STPPCLK**: Stop PHY clock
The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

### 62.14.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1)

This register is available in host and device modes.

**Address offset**: 0xE04

**Reset value**: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:4</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>RAMGATEEN</td>
<td>Enable RAM clock gating</td>
</tr>
<tr>
<td>30</td>
<td>Enable gating of the FIFO RAM.</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>CNTGATECLK[1:0]</td>
<td>Counter for clock gating</td>
</tr>
<tr>
<td>28</td>
<td>Indicates to the controller how many PHY Clock cycles and AHB Clock cycles of 'IDLE' (no activity) the controller waits for before Gating the respective PHY and AHB clocks internal to the controller.</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>00: 64 clocks</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>01: 128 clocks</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>10: Reserved</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>11: Reserved</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>GATEEN</td>
<td>Enable active clock gating</td>
</tr>
<tr>
<td>22</td>
<td>The application programs GATEEN to enable Active Clock Gating feature for the PHY and AHB clocks.</td>
<td></td>
</tr>
</tbody>
</table>
## 62.14.63 OTG_HS register map

The table below gives the USB OTG register map and reset values.

### Table 675. OTG_FS/OTG_HS register map and reset values

| Offset | Register name                  | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|--------------------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x000  | OTG_GOTGCTL                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x004  | OTG_GOTGINT                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 0   |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
| 0x008  | OTG_GAHBCFG                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x00C  | OTG_GUSBCFG                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 1   | 0   | 1   | 1   | 0   | 0   | 0   | 0   | 0   |
| 0x010  | OTG_GRSTCTL                    |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 1   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x014  | OTG_GINTSTS (Device mode)      |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0x014  | OTG_GINTSTS (Host mode)        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        | Reset value                    | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

*Reset value 000001 000000*
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Device mode</th>
<th>Host mode</th>
<th>Device mode</th>
<th>Host mode</th>
<th>Device mode</th>
<th>Host mode</th>
<th>Device mode</th>
<th>Host mode</th>
<th>Device mode</th>
<th>Host mode</th>
<th>Device mode</th>
<th>Host mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>OTG_GINTMSK</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
</tr>
<tr>
<td>0x01</td>
<td>OTG_GRXSTSR</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
</tr>
<tr>
<td>0x02</td>
<td>OTG_GRXSTSP</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
</tr>
<tr>
<td>0x03</td>
<td>OTG_GCCFG</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
<td>0000000000</td>
</tr>
</tbody>
</table>
Table 675. OTG_FS/OTG_HS register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03C</td>
<td>OTG_CID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x054</td>
<td>OTG_GLPMCFG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>OTG_HPTXFSIZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x104</td>
<td>OTG_DIEPTXF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x120</td>
<td>OTG_DIEPTXF7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x400</td>
<td>OTG_HCFG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x404</td>
<td>OTG_HFIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x408</td>
<td>OTG_HFNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x410</td>
<td>OTG_HPTXSTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x414</td>
<td>OTG_HAINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x418</td>
<td>OTG_HAINTMSK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x440</td>
<td>OTG_HPRT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x500</td>
<td>OTG_HCHAR0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03C</td>
<td>OTG_CID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x054</td>
<td>OTG_GLPMCFG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x100</td>
<td>OTG_HPTXFSIZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x104</td>
<td>OTG_DIEPTXF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x120</td>
<td>OTG_DIEPTXF7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x400</td>
<td>OTG_HCFG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x404</td>
<td>OTG_HFIR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x408</td>
<td>OTG_HFNUM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x410</td>
<td>OTG_HPTXSTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x414</td>
<td>OTG_HAINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x418</td>
<td>OTG_HAINTMSK</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x440</td>
<td>OTG_HPRT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x500</td>
<td>OTG_HCHAR0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Offset Register name</td>
<td>Reset value</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>0x504</td>
<td>OTG_HCSPLT0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x508</td>
<td>OTG_HCINT0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x508</td>
<td>OTG_HCINT0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x50C</td>
<td>OTG_HCINTMŠK0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x510</td>
<td>OTG_HCTSIZ0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x514</td>
<td>OTG_HCDMA0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x6E0</td>
<td>OTG_HCCHAR15</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x6E4</td>
<td>OTG_HCSPLT15</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x6E6</td>
<td>OTG_HCINT15</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0x6EC</td>
<td>OTG_HCINTMŠK15</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 675. OTG_FS/OTG_HS register map and reset values (continued)

<p>| Offset  | Register name       | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x6F0   | OTG_HCTSIZ15        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x6F4   | OTG_HCDMA15         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x800   | OTG_DCFG            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x804   | OTG_DCTL            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x808   | OTG_DSTS            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x810   | OTG_DIEPMSK         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x814   | OTG_DOEPMASK        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x818   | OTG_DINT            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x81C   | OTG_DINTMSK         |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x830   | OTG_DTHRCCTL        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x834   | OTG_DIEPMPMSK       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value         | 0  | 0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |</p>
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x900</td>
<td>OTG_DIEPCTL0 (INT/BULK)</td>
<td>0x908</td>
<td>OTG_DIEPINT0</td>
</tr>
<tr>
<td>Reset value</td>
<td>0x900</td>
<td>Reset value</td>
<td>0x908</td>
</tr>
<tr>
<td>0x908</td>
<td>OTG_DIEPINT0</td>
<td>0x908</td>
<td>OTG_DIEPINT0</td>
</tr>
<tr>
<td>Reset value</td>
<td>0x908</td>
<td>Reset value</td>
<td>0x908</td>
</tr>
<tr>
<td>0x910</td>
<td>OTG_DIEPTSIZ0</td>
<td>0x910</td>
<td>OTG_DIEPTSIZ0</td>
</tr>
<tr>
<td>Reset value</td>
<td>0x910</td>
<td>Reset value</td>
<td>0x910</td>
</tr>
<tr>
<td>0x914</td>
<td>OTG_DIEPTSIZ0</td>
<td>0x914</td>
<td>OTG_DIEPTSIZ0</td>
</tr>
<tr>
<td>Reset value</td>
<td>0x914</td>
<td>Reset value</td>
<td>0x914</td>
</tr>
<tr>
<td>0x918</td>
<td>OTG_DTXFSTS0</td>
<td>0x918</td>
<td>OTG_DTXFSTS0</td>
</tr>
<tr>
<td>Reset value</td>
<td>0x918</td>
<td>Reset value</td>
<td>0x918</td>
</tr>
</tbody>
</table>

Table 675. OTG_FS/OTG_HS register map and reset values (continued)
Table 675. OTG_FS/OTG_HS register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xB00</td>
<td>OTG_DOEPCCTL0</td>
<td>0xB08</td>
<td>OTG_DOEPIINT0</td>
<td>0xB10</td>
<td>OTG_DOEPTSIZ0</td>
<td>0xB14</td>
<td>OTG_DOEPDMA0</td>
<td>0xB20</td>
<td>OTG_DOEPCCTL1</td>
<td>0xB28</td>
<td>OTG_DOEPIINT1</td>
<td>0xB30</td>
<td>OTG_DOEPTSIZ1</td>
<td>0xB34</td>
<td>OTG_DOEPDMA1</td>
<td>Iterating preceding block of registers starting at offset 0xB20.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>EPENA (EPDIS)</td>
<td></td>
<td>STU (PKTCTRL)</td>
<td></td>
<td>Res (Res)</td>
<td></td>
<td>RPDMAADDR</td>
<td></td>
<td>EPENA (EPDIS)</td>
<td></td>
<td>STU (PKTCTRL)</td>
<td></td>
<td>RPDMAADDR</td>
<td></td>
<td>EPENA (EPDIS)</td>
<td></td>
<td>STU (PKTCTRL)</td>
<td></td>
<td>RPDMAADDR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>EPO (EPO)</td>
<td></td>
<td>Res (Res)</td>
<td></td>
<td>XFRSIZ (XFRSIZ)</td>
<td></td>
<td>Res (Res)</td>
<td></td>
<td>EPO (EPO)</td>
<td></td>
<td>Res (Res)</td>
<td></td>
<td>XFRSIZ (XFRSIZ)</td>
<td></td>
<td>EPO (EPO)</td>
<td></td>
<td>Res (Res)</td>
<td></td>
<td>XFRSIZ (XFRSIZ)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Reset value rows indicate initial state of registers after reset.
Table 675. OTG_FS/OTG_HS register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xC00  | OTG_DOEPCTL8  | EPENA | EPDIS | SOF| S0PD | SNAK | CNAK | STALL | SNPM | EP TYP | EPENA | EPDIS | SOF | S0PD | SNAK | CNAK | STALL | SNPM | EP TYP | NASTS | ORIG | USBAEP | USD | MPSIZ |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xC08  | OTG_DOEPINT8  | EPENA | EPDIS | SOF | S0PD | SNAK | CNAK | STALL | SNPM | EP TYP | EPENA | EPDIS | SOF | S0PD | SNAK | CNAK | STALL | SNPM | EP TYP | NASTS | ORIG | USBAEP | USD | MPSIZ |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xC10  | OTG_DOEPTSIZ8 | RXDPID | STUPCNT | PKTCNT | XFRSIZ |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xC14  | OTG_DOEPDMA8  | DMAADDR |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xE00  | OTG_PCGCCTL   | SUSP | SLEEP | ENLONG | PHYSUSP | GATEHCLK | STPCLK | GATEN |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xE04  | OTG_PCGCCTL1  | RAMAGATEEN | ONGATEEN | GATEEN |
|        | Reset value   | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Refer to Section 2.3 on page 149 for the register boundary addresses.
62.15 OTG_HS programming model

62.15.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG_HS controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG_HS controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core’s configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   - Global interrupt mask bit GINTMSK = 1
   - Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   - Periodic Tx FIFO empty level

2. Program the following fields in the OTG_GUSBCFG register:
   - OTG_HS timeout calibration field
   - USB turnaround time field

3. The software must unmask the following bits in the OTG_GINTMSK register:
   - OTG interrupt mask
   - Mode mismatch interrupt mask

4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG_HS controller is operating in host or device mode.
62.15.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. The port is now considered powered however this does not actually drive VBUS on the USB, so a further action must be taken to control the external circuitry in the system so as to enable VBUS generation.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.
62.15.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval

2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable thresholding.

3. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.

4. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF

5. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.

6. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS register to determine the enumeration speed and perform the steps listed in Endpoint initialization on enumeration completion on page 3199.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

62.15.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (OTG_HCDMAx register in host mode and OTG_DIEPDMAx/OTG_DOEPDMAx register in peripheral mode) to access the data buffers.

62.15.5 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:
1. Program the OTG_GINTMSK register to unmask the following:

2. Channel interrupt
   - Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   - Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).

3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.

4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.

5. Program the selected channel’s OTG_HCTSiZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).

6. Program the OTG_HCCHARx register of the selected channel with the device’s endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).

7. Program the selected channels in the OTG_HCSPLTx register(s) with the hub and port addresses (split transactions only).

8. Program the selected channels in the OTG_HCDMAx register(s) with the buffer start address (DMA transactions only).

### Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG_HS host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG_HS host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG_HS host checks for space to write the disable request on the disabled channel’s turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in OTG_HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:
1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.

2. When an XFRC interrupt in OTG_HCINTx is received during a non periodic IN transfer or high-bandwidth interrupt IN transfer.

3. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).

4. When the application aborts a transfer before normal completion.

**Ping protocol**

When the OTG_HS host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints. The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the OTG_HS host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO). This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in OTG_HCTSIZx before enabling the channel or by just writing the OTG_HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the OTG_HS host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to set the DOPING bit in OTG_HCTSIZx for a NAK/NYET response in case of bulk/control OUT. The OTG_HS host automatically sets the DOPING bit in OTG_HCTSIZx, and issues the ping tokens for bulk/control OUT. The OTG_HS host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

**Operational model**

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**
  The OTG_HS host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG_HS host determines the actual packet size based on the programmed maximum packet size and transfer size.
• **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (0x0010).
• **Bulk and control OUT/SETUP transactions**

A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in Figure 924. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

- The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
- The non-periodic transmit FIFO can hold two packets (1 Kbyte for HS).
- The non-periodic request queue depth = 4.

• **Normal bulk and control OUT/SETUP operations**

The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses
Figure 924. Normal bulk/control OUT/SETUP

1. The grayed elements are not relevant in the context of this figure.
The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**
  a) Bulk/control OUT/SETUP

```c
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else if (STALL)
{
    Transfer Done = 1
    Unmask CHH
    Disable Channel
}
else if (NAK or TXERR )
{
    Rewind Buffer Pointers
    Unmask CHH
    Disable Channel
    if (TXERR)
    {
        Increment Error Count
        Unmask ACK
    }
    else
    {
        Reset Error Count
    }
}
else if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
```
else if (ACK)
{
    Reset Error Count
    Mask ACK
}

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

Unmask (TXERR/XFRC/BBERR/STALL/DERR)
if (XFRC)
{
    Reset Error Count
    Unmask CHH
    Disable Channel
    Reset Error Count
    Mask ACK
}
else if (TXERR or BBERR or STALL)
{
    Unmask CHH
    Disable Channel
    if (TXERR)
    {
        Increment Error Count
        Unmask ACK
    }
}
else if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel
    }
}
else if (ACK)
{
    Reset Error Count
    Mask ACK
}
else if (DTERR)
{
    Reset Error Count
}

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

  A typical bulk or control IN pipelined transaction-level operation is shown in Figure 925. See channel 2 (ch_2). The assumptions are:
  
  - The application is attempting to receive two maximum-packet-size packets (transfer size = 1024 bytes).
  - The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (520 bytes for HS).
  - The non-periodic request queue depth = 4.
Figure 925. Bulk/control IN transactions

1. The grayed elements are not relevant in the context of this figure.
The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

• Control transactions

Setup, data, and status stages of a control transfer must be performed as three separate transfers. Setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in
OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 926*. The assumptions are:
  - The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)
  - The periodic transmit FIFO can hold one packet (1 Kbyte)
  - Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_HS host writes an entry to the periodic request queue.
4. The OTG_HS host attempts to send an OUT token in the next (odd) frame.
5. The OTG_HS host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**
  a) **Interrupt OUT**

  Unmask (NAK/TXERR/STALL/XFRC/FRMOR)

---

**Figure 926. Normal interrupt OUT**

Host

<table>
<thead>
<tr>
<th>Application</th>
<th>AHB</th>
<th>Host</th>
<th>USB</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>init_reg(ch_2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set_ch_en (ch_2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>write_tx_fifo (ch_1)</td>
<td>MPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>init_reg(ch_1)</td>
<td>MPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>read_rx_fifo</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>read_rx_sts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>init_reg(ch_2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>set_ch_en (ch_2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>init_reg(ch_1)</td>
<td>MPS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>write_tx_fifo (ch_1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
if (XFRC)
{
    Reset Error Count
    Mask ACK
    De-allocate Channel
}
else
if (STALL or FRMOR)
{
    Mask ACK
    Unmask CHH
    Disable Channel
    if (STALL)
    {
        Transfer Done = 1
    }
}
else
if (NAK or TXERR)
{
    Rewind Buffer Pointers
    Reset Error Count
    Mask ACK
    Unmask CHH
    Disable Channel
}
else
if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
}
else
{
    Re-initialize Channel (in next b_interval - 1 Frame)
}
else
if (ACK)
{
    Reset Error Count
    Mask ACK
}
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

**Interrupt IN**

Unmask (NAK/ TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)

if (XFRC)
{
  Reset Error Count
  Mask ACK
  if (OTG_HCTSIZx.PKTCNT == 0)
  {
    De-allocate Channel
  }
  else
  {
    Transfer Done = 1
    Unmask CHH
    Disable Channel
  }
}

else
if (STALL or FRMOR or NAK or DTERR or BBERR)
{
  Mask ACK
  Unmask CHH
  Disable Channel
  if (STALL or BBERR)
  {
    Reset Error Count
    Transfer Done = 1
  }
  else
  if (!FRMOR)
  {
    Reset Error Count
  }
}
else
if (TXERR)
{
  Increment Error Count
  Unmask ACK
  Unmask CHH
  Disable Channel
}
else
if (CHH)
{
    Mask CHH
    if (Transfer Done or (Error_count == 3))
    {
        De-allocate Channel
    }
    else
    {
        Re-initialize Channel (in next b_interval - 1 /Frame)
    }
}
else
if (ACK)
{
    Reset Error Count
    Mask ACK
}

• Interrupt IN transactions
  The assumptions are:
  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).
  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).
  – Periodic request queue depth = 4.

• Normal interrupt IN operation
  The sequence of operations is as follows:
  1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
  2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
  3. The OTG_HS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
  4. The OTG_HS host attempts to send an IN token in the next (odd) frame.
  5. As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.
  6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
  7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
  8. The core generates an XFRC interrupt as soon as the receive packet status is read.
  9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-
initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.
1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in Figure 928. The assumptions are:
  
  - The application is attempting to send one packet every frame (up to 1 maximum
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 Kbyte).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG_HS host writes an entry to the periodic request queue.
4. The OTG_HS host attempts to send the OUT token in the next frame (odd).
5. The OTG_HS host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses
Figure 928. Isochronous OUT transactions

1. The grayed elements are not relevant in the context of this figure.

- Interrupt service routine for isochronous OUT/IN transactions

Code sample: isochronous OUT

Unmask (FRMOR/XFRC)

if (XFRC)
{  
  De-allocate Channel  
}  
else  
  if (FRMOR)  
  {  
    Unmask CHH  
    Disable Channel  
  }  
else  
  if (CHH)  
  {  
    Mask CHH  
    De-allocate Channel  
  }  

Code sample: Isochronous IN  
Unmask (TXERR/XFRC/FRMOR/BBERR)  
if (XFRC or FRMOR)  
{  
  if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))  
  {  
    Reset Error Count  
    De-allocate Channel  
  }  
else  
  {  
    Unmask CHH  
    Disable Channel  
  }  
}  
else  
  if (TXERR or BBERR)  
  {  
    Increment Error Count  
    Unmask CHH  
    Disable Channel  
  }  
else  
  if (CHH)  
  {  
    Mask CHH  
    if (Transfer Done or (Error_count == 3))  
    {  
      De-allocate Channel  
    }  
  }
else
{
    Re-initialize Channel
}

- **Isochronous IN transactions**
  The assumptions are:
  - The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).
  - The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).
  - Periodic request queue depth = 4.

  The sequence of operations is as follows:
  1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
  2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
  3. The OTG_HS host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
  4. The OTG_HS host attempts to send an IN token in the next odd frame.
  5. As soon as the IN packet is received and written to the receive FIFO, the OTG_HS host generates an RXFLVL interrupt.
  6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
  7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
  8. The core generates an XFRC interrupt as soon as the receive packet status is read.
  9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.
Figure 929. Isochronous IN transactions

1. The grayed elements are not relevant in the context of this figure.

**Selecting the queue depth**

Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

The non-periodic request queue depth affects the performance of non-periodic
transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core’s periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**
  
  OTG_HS controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG_HS controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

  When OTG_HS controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**
  
  The sequence of operations is as follows:

  1. Initialize and enable channel 1 as explained in Section: Channel initialization.
  2. The OTG_HS host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG_HS host uses the programmed DMA address to fetch the packet.
  3. After fetching the last 32-bit word of the second (last) packet, the OTG_HS host masks channel 1 internally for further arbitration.
  4. The OTG_HS host generates a CHH interrupt as soon as the last packet is sent.
  5. In response to the CHH interrupt, de-allocate the channel for other transfers.
**NAK and NYET handling with internal DMA:**

1. The OTG_HS host sends a bulk OUT transaction.
2. The device responds with NAK or NYET.
3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.
4. The core automatically issues a ping token.
5. When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.
The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**
  The sequence of operations is as follows:
  1. Initialize and enable the used channel (channel x) as explained in Section: Channel initialization.
  2. The OTG_HS host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).
  3. The OTG_HS host starts writing the received data to the system memory as soon as the last byte is received with no errors.
  4. When the last packet is received, the OTG_HS host sets an internal flag to remove any extra IN requests from the request queue.
  5. The OTG_HS host flushes the extra requests.
  6. The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.
  7. The OTG_HS host generates the CHH interrupt as soon as the disable request comes to the top of the queue.
  8. In response to the CHH interrupt, de-allocate the channel for other transfers.
Interrupt OUT transactions in DMA mode

1. Initialize and enable channel x as explained in Section : Channel initialization.
2. The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The OTG_HS host attempts to send the OUT token at the beginning of the next odd HS frame/micro-frame.
4. After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

**Figure 932. Normal interrupt OUT transactions - DMA mode**

- **Interrupt IN transactions in DMA mode**
  The sequence of operations (channelx) is as follows:
  1. Initialize and enable channel x as explained in *Section: Channel initialization*.
  2. The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host writes consecutive writes up to MC times.
3. The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.
4. As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

**Figure 93. Normal interrupt IN transactions - DMA mode**

- **Isochronous OUT transactions in DMA mode**
  1. Initialize and enable channel x as explained in Section: Channel initialization.
  2. The OTG_HS host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth
transfers, the OTG_HS host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

3. The OTG_HS host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.

4. After successfully transmitting the packet, the OTG_HS host generates a CHH interrupt.

5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

**Figure 934. Normal isochronous OUT transaction - DMA mode**

- **Isochronous IN transactions in DMA mode**
  
  The sequence of operations ((channel x) is as follows:
  
  1. Initialize and enable channel x as explained in *Section: Channel initialization*.
  2. The OTG_HS host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG_HS host performs consecutive write operations up to MC times.
3. The OTG_HS host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.
4. As soon the packet is received and written to the receive FIFO, the OTG_HS host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

![Figure 935. Normal isochronous IN transactions - DMA mode](image)

- **Bulk and control OUT/SETUP split transactions in DMA mode**
  The sequence of operations in (channel x) is as follows:
  1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
  2. The OTG_HS host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch.
  3. After successfully transmitting start split, the OTG_HS host generates the CHH interrupt.
  4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
5. After successfully transmitting complete split, the OTG_HS host generates the CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

**Bulk/control IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x as explained in [Section: Channel initialization].
2. The OTG_HS host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG_HS host masks the channel x internally for the arbitration after writing the request.
3. As soon as the IN token is transmitted, the OTG_HS host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 and re-enable the channel to send the complete split token. This unmask channel x for arbitration.
5. The OTG_HS host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.
6. The OTG_HS host starts writing the packet to the system memory after receiving the packet successfully.
7. As soon as the received packet is written to the system memory, the OTG_HS host generates a CHH interrupt.
8. In response to the CHH interrupt, de-allocate the channel.

**Interrupt OUT split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel 1 for start split as explained in [Section: Channel initialization]. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. The OTG_HS host starts reading the packet.
3. The OTG_HS host attempts to send the start split transaction.
4. After successfully transmitting the start split, the OTG_HS host generates the CHH interrupt.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
6. After successfully completing the complete split transaction, the OTG_HS host generates the CHH interrupt.
7. In response to CHH interrupt, de-allocate the channel.

**Interrupt IN split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in [Section: Channel initialization].
2. The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.
6. As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory.

7. The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory.

8. In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- **Isochronous OUT split transactions in DMA mode**
  The sequence of operations (channel x) is as follows:
  1. Initialize and enable channel x for start split (begin) as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1. Program the MPS field.
  2. The OTG_HS host starts reading the packet.
  3. After successfully transmitting the start split (begin), the OTG_HS host generates the CHH interrupt.
  4. In response to the CHH interrupt, reinitialize the registers to send the start split (end).
  5. After successfully transmitting the start split (end), the OTG_HS host generates a CHH interrupt.
  6. In response to the CHH interrupt, de-allocate the channel.

- **Isochronous IN split transactions in DMA mode**
  The sequence of operations (channel x) is as follows:
  1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
  2. The OTG_HS host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
  3. The OTG_HS host attempts to send the start split IN token at the beginning of the next odd micro-frame.
  4. The OTG_HS host generates the CHH interrupt after successfully transmitting the start split IN token.
  5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.
  6. As soon as the packet is received successfully, the OTG_HS host starts writing the data to the system memory. The OTG_HS host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.
62.15.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   – SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   – INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   – OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   – STUPM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DIEPMSK
   – TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   – Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   – Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   – STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. For USB OTG_HS in DMA mode, the OTG_DOEPDMA0 register must have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. For USB OTG_HS in DMA mode, program the OTG_DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet
Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.

2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.

3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.

4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.

5. Set up the data FIFO RAM for each FIFO.

6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number

2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).

2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:

NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.
Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

- Packet read

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).

2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.

3. If the received packet’s byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.

4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
      PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0), DPID = (0b00).
      These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
      PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num, DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
      PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num, DPID = (0b00).
      These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
      PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
      PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
      These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.

5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.

6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

*Figure 936* provides a flowchart of the above procedure.
SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**
  1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
    - STUPCNT = 3 in OTG_DOEPTSIZx
  2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
    - The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
    - 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the
receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

- **Internal data flow**
  1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
     - The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
  2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
     - The first word contains control information used internally by the core
     - The second word contains the first 4 bytes of the SETUP command
     - The third word contains the last 4 bytes of the SETUP command
  3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
  4. On the AHB side, SETUP packets are emptied by the application.
  5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
  6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**
  1. Program the OTG_DOEPTSIZx register.
     - STUPCNT = 3
  2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
  3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
     - On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.
Handling more than three back-to-back SETUP packets

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG_HS controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

Setting the global OUT NAK

Internal data flow:
1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets.
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:
1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   - SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   - GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   - CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   - GONAKEFFM = 1 in OTG_GINTMSK

**Disabling an OUT endpoint**
The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:
1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   - SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   - EPDIS = 1 in OTG_DOEPCTLx
   - SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   - EPDIS = 0 in OTG_DOEPCTLx
   - EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   - SGONAK = 0 in OTG_DCTL

**Transfer Stop Programming for OUT endpoints**
The application must use the following programming sequence to stop any transfers (because of an interrupt from the host, typically a reset).

**Sequence of operations:**
1. Enable all OUT endpoints by setting
   – EPENA = 1 in all OTG_DOEPCTLx registers.

2. Flush the RxFIFO as follows
   – Poll OTG_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on OTG_GRSTCTL.RXFFLSH = 1
   – Poll OTG_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than
     10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen
     before the timeout, then the RxFIFO flush is successful. If at the moment the
     timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from
     the host) then go back (once only) to the previous step ("Perform read modify write
     operation").

3. Before disabling any OUT endpoint, the application must enable Global OUT NAK
   mode in the core, according to the instructions in “Setting the global OUT NAK on
   page 3204”. This ensures that data in the RxFIFO is sent to the application
   successfully. Set SGONAK = 1 in OTG_DCTL

4. Wait for the GONAKEFF interrupt (OTG_GINTSTS)

5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers OTG_DOEPCTLx
   – SNAK = 1 in registers OTG_DOEPCTLx

6. Wait for the EPDIS interrupt in OTG_DOEPINTx for each OUT endpoint programmed
   in the previous step. The EPDIS interrupt in OTG_DOEPINTx indicates that the
   corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is
   asserted, the following bits are cleared:
   – EPENA = 0 in registers OTG_DOEPCTLx
   – EPDIS = 0 in registers OTG_DOEPCTLx
   – SNAK = 0 in registers OTG_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or
interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory
   to accommodate all data to be received as part of the OUT transfer.

2. For OUT transfers, the transfer size field in the endpoint’s transfer size register must be
   a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – packet count[EPNUM] = n
   – n > 0

3. On any OUT endpoint interrupt, the application must read the endpoint's transfer size
   register to calculate the size of the payload in the memory. The received payload size
   can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core
     updated final transfer size
   – Number of USB packets in which this payload was received = application
     programmed initial packet count – core updated final packet count
Internal data flow:
1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, resends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:
1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   - EPENA = 1 in OTG_DOEPCTLx
   - CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   - This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

  This section describes a regular isochronous OUT data transfer.

  **Application requirements:**
  1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
  2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
  3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
  4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

  **Internal data flow:**
  1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
  2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
     - EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
  3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

  **Application programming sequence:**
1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   - EPENA = 1
   - CNAK = 1
   - EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   - This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPSISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   - RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   - RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2
   - RXDPID = D2 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 3
   The number of USB packets in which this payload was received = Application programmed initial packet count – core updated final packet count
   The application can discard invalid data packets.

- Incomplete isochronous OUT data transfers

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:
1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application may fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
   - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   - When the isochronous OUT data packet is received with CRC errors
   - When the isochronous OUT token received by the core is corrupted
   - When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPSISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At
this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPSIOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.

2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   - When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.

3. When it receives an INCOMPSIOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   - EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   - EPENA = 1 (in OTG_DOEPCTLx)

4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.

5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.

6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   - Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

• **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.

2. Disable the required endpoint
   - When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).
     The STALL bit always takes precedence over the NAK bit.

3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.

4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

**Examples**

This section describes and depicts some fundamental transfer types and scenarios.

• Bulk OUT transaction
Figure 938 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

**Figure 938. Bulk OUT transaction**

After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

**IN data transfers**

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.
1. The application can either choose the polling or the interrupt mode.
   – In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTx register, to determine if there is enough space in the data FIFO.
   – In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTx register, to determine if there is enough space in the data FIFO.
   – To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   – To write zero length packet, the application must not look at the FIFO space.

2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:
1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint’s transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:
1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNNE interrupt (in OTG_DIEPINTx).
6. If the application masked this interrupt earlier, it must be unmasked as follows:
   - INEPNEM = 1 in OTG_DIEPMSK

**IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:
1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   - SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   - EPDIS = 1 in OTG_DIEPCTLx
   - SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   - EPENA = 0 in OTG_DIEPCTLx
   - EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   - TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   - TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

**Transfer Stop Programming for IN endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).
Sequence of operations:

1. Disable the IN endpoint by setting:
   - EPDIS = 1 in all OTG_DIEPCTLx registers
2. Wait for the EPDIS interrupt in OTG_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   - EPDIS = 0 in OTG_DIEPCTLx
   - EPENA = 0 in OTG_DIEPCTLx
3. Flush the TxFIFO by programming the following bits:
   - TXFFLSH = 1 in OTG_GRSTCTL
   - TXFNUM = “FIFO number specific to endpoint” in OTG_GRSTCTL
4. The application can start polling till TXFFLSH in OTG_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

Generic non-periodic IN data transfers

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
     Transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = x
   - To transmit a single zero-length data packet:
     Transfer size[EPNUM] = 0
     Packet count[EPNUM] = 1
   - To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
     First transfer: transfer size[EPNUM] = x × MPSIZ[epnum]; packet count = n;
     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   - Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)
Internal data flow:
1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:
1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

• Generic periodic IN data transfers

This section describes a typical periodic IN data transfer.

Application requirements:
1. Application requirements 1, 2, 3, and 4 of Generic non-periodic IN data transfers on page 3214 also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To
transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

\[
\text{transfer size}[\text{EPNUM}] = x \times \text{MPSIZ}[\text{EPNUM}] + \text{sp}
\]

(where \(x\) is an integer \(\geq 0\), and \(0 \leq \text{sp} < \text{MPSIZ}[\text{EPNUM}]\))

If (\(\text{sp} > 0\)), packet count[EPNUM] = \(x + 1\)

Otherwise, packet count[EPNUM] = \(x\);

\[
\text{MCNT}[\text{EPNUM}] = \text{packet count}[\text{EPNUM}]
\]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:

\[
\text{transfer size}[\text{EPNUM}] = 0
\]

\[
\text{packet count}[\text{EPNUM}] = 1
\]

\[
\text{MCNT}[\text{EPNUM}] = \text{packet count}[\text{EPNUM}]
\]

2. The application can only schedule data transfers one frame at a time.

\[
(\text{MCNT} – 1) \times \text{MPSIZ} \leq \text{XFERSIZ} \leq \text{MCNT} \times \text{MPSIZ}
\]

– PKTCNT = MCNT (in OTG_DIEPTSIZx)

– If \(\text{XFERSIZ} < \text{MCNT} \times \text{MPSIZ}\), the last data packet of the transfer is a short packet.

– Note that: \(\text{MCNT}\) is in OTG_DIEPTSIZx, \(\text{MPSIZ}\) is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and Xfersiz is in OTG_DIEPTSIZx

3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:

– A zero data length packet would be transmitted on the USB for isochronous IN endpoints

– A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.

2. The application must also write the required data to the associated transmit FIFO for the endpoint.

3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.

4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO
mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.

− A zero-length data packet is transmitted on the USB for isochronous IN endpoints
− A NAK handshake is transmitted on the USB for interrupt IN endpoints

5. The packet count for the endpoint is decremented by 1 under the following conditions:

− For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
− For interrupt endpoints, when an ACK handshake is transmitted
− When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.

6. At the “Periodic frame Interval” (controlled by PFiVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

• Incomplete isochronous IN data transfers

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it
eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.

3. The application must set the NAK bit and the disable bit for the endpoint.

4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).

2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.

3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.

4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.

5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   - SNAK = 1 in OTG_DIEPCTLx
   - EPDIS = 1 in OTG_DIEPCTLx

6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

  This section describes how the application can stall a non-isochronous endpoint.

  Application programming sequence:
1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   - STALL = 1 in OTG_DIEPCTLx
   - The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the
   application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on
   the endpoint type. In case of a non-periodic endpoint, the application must re-enable
   the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the
   STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a
   SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the
   STALL bit must be set or cleared before the application sets up the status stage
   transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host
sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the
application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt
in OTG_DOEPINTx during the data stage of the control transfer, after the core has
transferred the amount of data specified in the SETUP packet. Then, when the application
receives this interrupt, it must set the STALL bit in the corresponding endpoint control
register, and clear this interrupt.

62.15.7 Worst case response time

When the OTG_HS controller acts as a device, there is a worst case response time for any
tokens that follow an isochronous OUT. This worst case response time depends on the AHB
clock frequency.

The core registers are in the AHB domain, and the core does not accept another token
before updating these register values. The worst case is for any token following an
isochronous OUT, because for an isochronous transaction, there is no handshake and the
next token may come sooner. This worst case value is 7 PHY clocks when the AHB clock is
the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK
and drops isochronous and SETUP tokens. The host interprets this as a timeout condition
for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete
isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer
interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were
dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY
clocks after it has received an IN token, to get the FIFO status, and thus the first data from
the PFC block. This time involves the synchronization delay between the PHY and AHB
clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock.
In this case, the delay is 5 clocks.
Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 939 has the following signals:
- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to Table 673: TRDT values.

**Figure 939. TRDT max timing case**

![TRDT max timing case diagram](image-url)
62.15.8 **OTG programming model**

The OTG_HS controller is an OTG device. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device.
63 USB Type-C®/USB Power Delivery interface (UCPD)

63.1 Introduction

The USB Type-C/USB Power Delivery interface complies with:
- Universal Serial Bus Type-C Cable and Connector Specification: release 2.3, Oct 2023
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.2, version 1.0, October 2023

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

63.2 UCPD main features

- Compliance with USB Type-C specification release 2.3
- Compliance with USB Power Delivery specifications revision 2.0 and 3.2
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - “Dead battery” Rd support
  - USB Power Delivery message transmission and reception
  - FRS (fast role swap) Rx support
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - FRS detection, generating an interrupt
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

63.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.
63.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), BIST test data (Tx and Rx), and Power Delivery Rx FRS signaling.

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX1/2 pins (alternate functions), external NMOS transistors that ensure low-resistance pull-down on CC lines.

The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).
In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages and FRS signaling, which allows low-power operation.

### 63.4.1 UCPD block diagram

![UCPD block diagram](image)

The following table lists external signals (alternate or additional I/O functions).

<table>
<thead>
<tr>
<th>Pin name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UCPDx.FRSTX1</td>
<td>Output</td>
<td>USB Type-C fast role swap (FRS) signaling, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the CC1 line.</td>
</tr>
<tr>
<td>UCPDx.FRSTX2</td>
<td>Output</td>
<td>USB Type-C fast role swap (FRS) signaling, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the CC2 line.</td>
</tr>
<tr>
<td>UCPDx.CC1</td>
<td>Input/output</td>
<td>USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.</td>
</tr>
<tr>
<td>UCPDx.CC2</td>
<td>Input/output</td>
<td>USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.</td>
</tr>
<tr>
<td>UCPDx.DBCC1</td>
<td>Input</td>
<td>USB Type-C configuration control line 1 dead battery signal, to be routed to the USB Type-C connector CC1 terminal if dead battery support is required.</td>
</tr>
<tr>
<td>UCPDx.DBCC2</td>
<td>Input</td>
<td>USB Type-C configuration control line 2 dead battery signal, to be routed to the USB Type-C connector CC2 terminal if dead battery support is required.</td>
</tr>
</tbody>
</table>
The following table lists key internal signals.

<table>
<thead>
<tr>
<th>Internal signal name</th>
<th>Signal type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ucpd_pclk</td>
<td>Input</td>
<td>APB clock for registers</td>
</tr>
<tr>
<td>ucpd_ker_ck</td>
<td>Input</td>
<td>Kernel clock</td>
</tr>
<tr>
<td>ucpd_tx_dma</td>
<td>Input/Output</td>
<td>Rx DMA acknowledge / request</td>
</tr>
<tr>
<td>ucpd_rx_dma</td>
<td>Input/Output</td>
<td>Tx DMA acknowledge / request</td>
</tr>
<tr>
<td>ucpd_it</td>
<td>Output</td>
<td>Interrupt request (all interrupts OR-ed) connected to NVIC</td>
</tr>
<tr>
<td>ucpd_wkup</td>
<td>Output</td>
<td>Wake-up request connected to EXTI</td>
</tr>
<tr>
<td>clk_rq</td>
<td>Output</td>
<td>Clock request connected to RCC</td>
</tr>
</tbody>
</table>

### 63.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck clock according to the PSC_UCPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 6 to 12 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

![Figure 941. Clock division and timing elements](image)

Refer to the USB PD specification in order to set appropriate delays. For $t_{TransitionWindow}$ and especially for $t_{InterFrameGap}$, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.
63.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification. On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

<table>
<thead>
<tr>
<th>Name</th>
<th>4b</th>
<th>5b</th>
<th>Symbol description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>11110</td>
<td>hex data 0</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>01001</td>
<td>hex data 1</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>10100</td>
<td>hex data 2</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>10101</td>
<td>hex data 3</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>01010</td>
<td>hex data 4</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>01011</td>
<td>hex data 5</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>01110</td>
<td>hex data 6</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>01111</td>
<td>hex data 7</td>
</tr>
<tr>
<td>8</td>
<td>1000</td>
<td>10010</td>
<td>hex data 8</td>
</tr>
<tr>
<td>9</td>
<td>1001</td>
<td>10011</td>
<td>hex data 9</td>
</tr>
<tr>
<td>A</td>
<td>1010</td>
<td>10110</td>
<td>hex data A</td>
</tr>
<tr>
<td>B</td>
<td>1011</td>
<td>10111</td>
<td>hex data B</td>
</tr>
<tr>
<td>C</td>
<td>1100</td>
<td>11010</td>
<td>hex data C</td>
</tr>
</tbody>
</table>
### Table 680. 4b5b symbol encoding table (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>4b</th>
<th>5b</th>
<th>Symbol description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1101</td>
<td>11011</td>
<td>hex data D</td>
</tr>
<tr>
<td>E</td>
<td>1110</td>
<td>11100</td>
<td>hex data E</td>
</tr>
<tr>
<td>F</td>
<td>1111</td>
<td>11101</td>
<td>hex data F</td>
</tr>
<tr>
<td>Sync-1</td>
<td>K-code</td>
<td>11000</td>
<td>Startsynch #1</td>
</tr>
<tr>
<td>Sync-2</td>
<td>K-code</td>
<td>10001</td>
<td>Startsynch #2</td>
</tr>
<tr>
<td>RST-1</td>
<td>K-code</td>
<td>00111</td>
<td>Hard Reset #1</td>
</tr>
<tr>
<td>RST-2</td>
<td>K-code</td>
<td>11001</td>
<td>Hard Reset #2</td>
</tr>
<tr>
<td>EOP</td>
<td>K-code</td>
<td>01101</td>
<td>EOP</td>
</tr>
<tr>
<td>Reserved Error</td>
<td>00000</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>00001</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>00010</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>00011</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>00100</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>00101</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Sync-3</td>
<td>K-code</td>
<td>00110</td>
<td>Startsynch #3</td>
</tr>
<tr>
<td>Reserved Error</td>
<td>01000</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>01100</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>10000</td>
<td>Do Not Use</td>
<td></td>
</tr>
<tr>
<td>Reserved Error</td>
<td>11111</td>
<td>Do Not Use</td>
<td></td>
</tr>
</tbody>
</table>
Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 942. K-code transmission

The following table lists the defined ordered sets, including all possible SOP* sequences. At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

Table 681. Ordered sets

<table>
<thead>
<tr>
<th>Ordered set name</th>
<th>K-code #1</th>
<th>K-code #2</th>
<th>K-code #3</th>
<th>K-code #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOP</td>
<td>Sync-1</td>
<td>Sync-1</td>
<td>Sync-1</td>
<td>Sync-2</td>
</tr>
<tr>
<td>SOP'</td>
<td>Sync-1</td>
<td>Sync-1</td>
<td>Sync-3</td>
<td>Sync-3</td>
</tr>
<tr>
<td>SOP''</td>
<td>Sync-1</td>
<td>Sync-3</td>
<td>Sync-1</td>
<td>Sync-3</td>
</tr>
<tr>
<td>Hard Reset</td>
<td>RST-1</td>
<td>RST-1</td>
<td>RST-1</td>
<td>RST-2</td>
</tr>
<tr>
<td>Cable Reset</td>
<td>RST-1</td>
<td>Sync-1</td>
<td>RST-1</td>
<td>Sync-3</td>
</tr>
<tr>
<td>SOP'_Debug</td>
<td>Sync-1</td>
<td>RST-2</td>
<td>RST-2</td>
<td>Sync-3</td>
</tr>
<tr>
<td>SOP''_Debug</td>
<td>Sync-1</td>
<td>RST-2</td>
<td>Sync-3</td>
<td>Sync-2</td>
</tr>
</tbody>
</table>

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 682. Validation of ordered sets

<table>
<thead>
<tr>
<th>Status</th>
<th>1st code</th>
<th>2nd code</th>
<th>3rd code</th>
<th>4th code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>Corrupt</td>
<td>K-code</td>
<td>K-code</td>
<td>K-code</td>
</tr>
<tr>
<td>Valid</td>
<td>K-code</td>
<td>Corrupt</td>
<td>K-code</td>
<td>K-code</td>
</tr>
</tbody>
</table>
Table 682. Validation of ordered sets (continued)

<table>
<thead>
<tr>
<th>Status</th>
<th>1st code</th>
<th>2nd code</th>
<th>3rd code</th>
<th>4th code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid</td>
<td>K-code</td>
<td>K-code</td>
<td>Corrupt</td>
<td>K-code</td>
</tr>
<tr>
<td>Valid (perfect)</td>
<td>K-code</td>
<td>K-code</td>
<td>K-code</td>
<td>K-code</td>
</tr>
<tr>
<td>Not valid (example)</td>
<td>K-code</td>
<td>Corrupt</td>
<td>K-code</td>
<td>Corrupt</td>
</tr>
</tbody>
</table>

**Bit ordering at transmission**

Allowed transmission data units / data sizes are in the following table.

**Table 683. Data size**

<table>
<thead>
<tr>
<th>Data unit</th>
<th>Non-encoded</th>
<th>Encoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>8-bits</td>
<td>10-bits</td>
</tr>
<tr>
<td>Word</td>
<td>16-bits</td>
<td>20-bits</td>
</tr>
<tr>
<td>DWord</td>
<td>32-bits</td>
<td>40-bits</td>
</tr>
</tbody>
</table>

The bit transmission order is shown in the following figure.

**Figure 943. Transmit order for various sizes of data**
Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

![Packet format](image)

**Figure 944. Packet format**

<table>
<thead>
<tr>
<th>Preamble (training for receiver)</th>
<th>SOP* (start of packet)</th>
<th>Header</th>
<th>Byte 0</th>
<th>Byte 1</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Byte n-1</td>
<td>Byte n</td>
<td>CRC</td>
<td>EOP</td>
<td></td>
</tr>
</tbody>
</table>

Legend:
- Provided by the physical layer, not 4b5b-encoded
- Provided by the physical layer, 4b5b-encoded
- Provided by the protocol layer, 4b5b-encoded

Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for InterFrameGap time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

![Line format of Hard Reset](image)

**Figure 945. Line format of Hard Reset**

<table>
<thead>
<tr>
<th>Preamble (training for receiver)</th>
<th>RST-1</th>
<th>RST-1</th>
<th>RST-1</th>
<th>RST-2</th>
</tr>
</thead>
</table>

Legend:
- Provided by the physical layer, not 4b5b-encoded
- Provided by the physical layer, 4b5b-encoded
Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

**Figure 946. Line format of Cable Reset**

<table>
<thead>
<tr>
<th>Preamble (training for receiver)</th>
<th>RST-1</th>
<th>Sync-1</th>
<th>RST-1</th>
<th>Sync-3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Legend:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provided by the physical layer, not 4b5b-encoded</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provided by the physical layer, 4b5b-encoded</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Collision avoidance

The physical layer respects the tInterFrameGap delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (nTransitionCount) transitions within tTransitionWindow (12 to 20 μs). The Power Delivery specification revision 3.1 also requires to manage the Rp value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.
The BIST test data packet frame format is shown in the following figure.

**Figure 947. BIST test data frame**

This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 944: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block’s behavior) that differentiates the general packet from the BIST Test Data packet.

**BIST Carrier Mode 2**

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

**Figure 948. BIST Carrier Mode 2 frame**

**63.4.4 UCPD BMC transmitter**

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

- \( t_{\text{InterFrameGap}} \): via IFRGAP (pre-defined value, can be altered)
- \( t_{\text{TransitionWindow}} \): via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.
63.4.5 UCPD BMC receiver

The UCPD BMC receiver performs:
- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

**Figure 950. UCPD BMC receiver architecture**

CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.
Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we open a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP’, and SOP’’), but also Hard Reset, Cable Reset, SOP’_Debug, SOP’’_Debug, and two extensions defined by registers UCPD_RX_ORDEXT1 and UCPD_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx message is arriving. When an “early EOP” is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one “UI” period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less than half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

63.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it still presents the “dead battery” Rd, provided that UCPDx_DBCC1 and UCPDx_DBCC2 pins are each connected to UCPDx_CC1 and UCPDx_CC2 pins, respectively.

If dead battery behavior is not required (for example for source only products), then UCPDx_DBCC1 and UCPDx_DBCC2 pins must both be tied to ground.
After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0] before setting the UCPD_DBDIS bit of the PWR_UCPDR register to remove dead battery pull-down resistor and allow the values just programmed to take effect.

Use of Standby low-power mode is possible for sinks in the unattached state.

63.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

63.4.8 UCPD fast role swap (FRS)

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the currently active I/O configured as UCPD_FRSTX1 (or 2) (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

FRS detection

FRS monitoring is enabled by setting the bit FRSRXEN, after writing PHYCCSEL that selects the active CC line depending on the cable orientation detected.

63.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1_TXDR and UCPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

63.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.
The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Event on the FRS detector (FRSEVT), hardware enable FRSRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

### 63.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   - On demand from protocol layer, send Tx message
   - Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

#### 63.5.1 Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.
3. Enable the analog Rx filter of either CC line, via the RXAFILTEN bit of the UCPD_CFGR2 register.

#### 63.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

<table>
<thead>
<tr>
<th>ANAMODE</th>
<th>ANASUBMODE[1:0]</th>
<th>Notes</th>
<th>TYPEC_VSTATE_CCx[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>00</td>
</tr>
<tr>
<td>0: Source</td>
<td>00: Disabled</td>
<td>Disabled</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>01: Default USB Rp</td>
<td>-</td>
<td>vRa[Def]</td>
</tr>
<tr>
<td></td>
<td>10: 1.5A Rp</td>
<td>-</td>
<td>vRa[1.5]</td>
</tr>
<tr>
<td>1: Sink</td>
<td>xx</td>
<td>-</td>
<td>vRa</td>
</tr>
</tbody>
</table>

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

**Note:** The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and
VBUS state (ON/OFF/+voltage level); discharge. UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls, to function correctly.

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.

2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).

3. In the case of no connection, wait for a connection event.

4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.

5. When a new interrupt/event occurs on PHY_EVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHY_EVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

- Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.

- Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).

- In the case of no connection, wait for a connection event.

- Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.

- When a new interrupt/event occurs on PHY_EVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHY_EVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):
### Table 685. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

<table>
<thead>
<tr>
<th>Type-C state</th>
<th>ANAMODE; ANASUBMO DE[1:0]</th>
<th>CCENABLE</th>
<th>PHYCCSEL</th>
<th>RDCH</th>
<th>CC[x] VCONN EN(1)</th>
<th>Event =&gt; go to next line</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unattached. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0 (don’t care)</td>
<td>00: [neither]</td>
<td>00: [neither]</td>
<td>Wait for sink attach detect ; seen on CC1 [EVT1]</td>
<td>Attachwait started (100-200 ms) ; now also see the Ra =&gt; requesting VCONN</td>
</tr>
<tr>
<td>Attachwait. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>01: CC2 disable (possible and recommended due to external VCONN switch)</td>
<td>0: [Normal]</td>
<td>10: [CC2 active]</td>
<td>Timer (100 ms) and no PHYEVT x</td>
<td>Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1</td>
</tr>
<tr>
<td>Attached. SRC [VCONN =&gt; CC2]</td>
<td>0:Source; 10:Rp1A5</td>
<td>[SinkTxOK]</td>
<td>1: [Rd on CC1]</td>
<td>10: [CC2 active]</td>
<td>00: [neither]</td>
<td>SW timers (SinkTxNG)</td>
<td>Source wants to initiate message sequence (SinkTxNG condition set first)</td>
</tr>
<tr>
<td>Unattached wait. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>1: [discharge]</td>
<td>&gt; 0.8V detection</td>
<td>00: [neither]</td>
<td>Discharge VCONN [CC2] actively [Rdch]; to &lt; 0.8V</td>
<td></td>
</tr>
<tr>
<td>Unattached. SRC</td>
<td>0:Source; 11:Rp3A0</td>
<td>11:both enabled</td>
<td>0: [Normal]</td>
<td>00: [neither]</td>
<td>00: [neither]</td>
<td></td>
<td>[Details as first line of table]</td>
</tr>
</tbody>
</table>

1. Two GPIOs to enable VCONN through external load switch components

#### 63.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.
 Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

**Hard Reset transmission**

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

**Use of DMA for transmission**

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:
- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

**63.5.4 USB PD receive**

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:
- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:
- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set
**Use of DMA for reception**

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:
- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

**63.5.5 UCPD software trimming**

The CC pull-up (Rp) and pull-down (Rd) devices must be trimmed on each part, to meet the required accuracy. The trimming values are saved in the non-volatile memory.

To trim the CC pull-up and pull-down devices by software, apply the following procedure:
1. Retrieve the trim values from the non-volatile memory (refer to Table 677: UCPD software trim data)
2. At initialization, write the trim values to the UCPD_CFG3 register bitfields as follows:
   - 3A0_CC1[3:0] to TRIM_CC1_RP[3:0]
   - 3A0_CC2[3:0] to TRIM_CC2_RP[3:0]
   - Rd_CC1[3:0] to TRIM_CC1_RD[3:0]
   - Rd_CC2[3:0] to TRIM_CC2_RD[3:0]
3. At each setting of ANASUBMODE to 1A5 or 3A0, respectively, write the trimming values to the UCPD_CFG3 register bitfields as follows:
   - 1A5_CC1[3:0] or 3A0_CC1[3:0], respectively, to TRIM_CC1_RP[3:0]
   - 1A5_CC2[3:0] or 3A0_CC2[3:0], respectively, to TRIM_CC2_RP[3:0]

**63.6 UCPD low-power modes**

A summary of low-power modes is shown below in Table 686: Effect of low power modes on the UCPD.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>No effect</td>
</tr>
<tr>
<td>Stop</td>
<td>Detection of events (Type-C, BMC Rx, FRS detection) remains operational and can wake up the MCU.</td>
</tr>
<tr>
<td>Standby</td>
<td>UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.</td>
</tr>
<tr>
<td>Unpowered</td>
<td>Dead battery pull-downs remain active.</td>
</tr>
</tbody>
</table>
The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later
- Activity on Power Delivery FRS detector which can potentially generate an FRS signaling detection event (that is, FRSEVT) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)
- FRS detection signal (from the selected CC line)

### 63.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Event flag/Interrupt clearing method</th>
<th>Interrupt enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRS detection</td>
<td>FRSEVT</td>
<td>Set FRSEVTCF</td>
<td>FRSEVTIE</td>
</tr>
<tr>
<td>Type C voltage level change on CC2</td>
<td>TYPECEVT2</td>
<td>Set TYPECEVT2CF</td>
<td>TYPECEVT2IE</td>
</tr>
<tr>
<td>Type C voltage level change on CC1</td>
<td>TYPECEVT1</td>
<td>Set TYPECEVT1CF</td>
<td>TYPECEVT1IE</td>
</tr>
<tr>
<td>Rx message received</td>
<td>RXMSGEND</td>
<td>Set RXMSGENDCF</td>
<td>RXMSGENDIE</td>
</tr>
<tr>
<td>Rx data overflow</td>
<td>RXOVR</td>
<td>Set RXOVRCF</td>
<td>RXOVR</td>
</tr>
<tr>
<td>Rx Hard Reset detected</td>
<td>RXHRSTDET</td>
<td>Set RXHRSTDETCF</td>
<td>RXHRSTDETIIE</td>
</tr>
<tr>
<td>Rx ordered set (4 K-codes) detected</td>
<td>RXORDDET</td>
<td>Set RXORDDETCF</td>
<td>RXORDDETI</td>
</tr>
<tr>
<td>Receive data register not empty</td>
<td>RXNE</td>
<td>Read data in UCPD_RXDR</td>
<td>RXNEIE</td>
</tr>
<tr>
<td>Tx data underrun</td>
<td>TXUND</td>
<td>Set TXUNDCF</td>
<td>TXUNDIE</td>
</tr>
<tr>
<td>Hard Reset sent</td>
<td>HRSTSENT</td>
<td>Set HRSTSENTCF</td>
<td>HRSTSENTIE</td>
</tr>
<tr>
<td>Hard Reset discarded</td>
<td>HRSTDISC</td>
<td>Set HRSTDISCCF</td>
<td>HRSTDISCIE</td>
</tr>
<tr>
<td>Transmit message aborted</td>
<td>TXMSGABT</td>
<td>Set TXMSGABTCF</td>
<td>TXMSGABTIE</td>
</tr>
</tbody>
</table>
When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register. Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

### 63.8 UCPD registers

#### 63.8.1 UCPD configuration register 1 (UCPD_CFGR1)

Address offset: 0x000  
Reset value: 0x0000 0000  
General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>UCPDEN</td>
</tr>
<tr>
<td>30</td>
<td>RXDMAEN</td>
</tr>
<tr>
<td>29</td>
<td>TXDMAEN</td>
</tr>
<tr>
<td>28</td>
<td>RXORDSETEN[8:0]</td>
</tr>
<tr>
<td>27</td>
<td>PSC_UCPDCLK[2:0]</td>
</tr>
</tbody>
</table>

Bit 31 **UCPDEN**: UCPD peripheral enable  
General enable of the UCPD peripheral.  
0: Disable  
1: Enable  
Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable  
When set, the bit enables DMA mode for reception.  
0: Disable  
1: Enable
Bit 29 **TXDMAEN**: Transmission DMA mode enable
When set, the bit enables DMA mode for transmission.
0: Disable
1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable
The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:

- 0bXXXXXXXX1: SOP detect enabled
- 0bXXXXXXXX1X: SOP' detect enabled
- 0bXXXXXXXX1XX: SOP'' detect enabled
- 0bXXXXXXX1XXX: Hard Reset detect enabled
- 0bXXXXXXX1XXXX: Cable Detect reset enabled
- 0bXXXXXXX1XXXXX: SOP_DEBUG enabled
- 0bXXXXXXX1XXXXXX: SOP''_Debug enabled
- 0bXXXXXXX1XXXXXXX: SOP extension#1 enabled
- 0bXXXXXXX1XXXXXXXX: SOP extension#2 enabled

Bits 19:17 **PSC_UCPDCLK[2:0]**: Pre-scaler division ratio for generating ucpd_clk
The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (ucpd_clk).

- 0x0: 1 (bypass)
- 0x1: 2
- 0x2: 4
- 0x3: 8
- 0x4: 16

It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range from 6 to 9 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 **TRANSWIN[4:0]**: Transition window duration
The bitfield determines the division ratio (the bitfield value minus one) of a hbit_clk divider producing $t_{TransitionWindow}$ interval.

- 0x00: Not supported
- 0x01: 2
- 0x09: 10 (recommended)
- 0x1F: 32

Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock
The bitfield determines the division ratio (the bitfield value minus one) of a ucpd_clk divider producing inter-frame gap timer clock ($t_{InterFrameGap}$).

- 0x00: Not supported
- 0x01: 2
- 0x0D: 14
- 0x0E: 15
- 0x0F: 16
- 0x1F: 32

The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.
Bits 5:0  **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock
   The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing half-bit clock (hbit_clk).
   0x00: 1 (bypass)
   0x1A: 27
   0x3F: 64

### 63.8.2 UCPD configuration register 2 (UCPD_CFGR2)

**Address offset**: 0x004

**Reset value**: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:9**  Reserved, must be kept at reset value.

**Bit 8**  **RXAFILTEN**: Rx analog filter enable
   Setting the bit enables the Rx analog filter required for optimum Power Delivery reception.
   0: Disable
   1: Enable

**Bits 7:4**  Reserved, must be kept at reset value.

**Bit 3**  **WUPEN**: Wake-up from Stop mode enable
   Setting the bit enables the UCPD_ASYNC_INT signal.
   0: Disable
   1: Enable

**Bit 2**  **FORCECLK**: Force CkReq clock request
   0: Do not force clock request
   1: Force clock request

**Bit 1**  **RXFILT2N3**: BMC decoder Rx pre-filter sampling method
   Number of consistent consecutive samples before confirming a new value.
   0: 3 samples
   1: 2 samples

**Bit 0**  **RXFILTDIS**: BMC decoder Rx pre-filter enable
   0: Enable
   1: Disable
   The sampling clock is that of the receiver (that is, after pre-scaler).
### 63.8.3 UCPD configuration register 3 (UCPD_CCFGR3)

- **Address offset:** 0x008
- **Reset value:** 0x0000 0000

Configuration of UCPD trimming of the CC pull-up and pull-down devices. The trimming is managed by hardware until the first software write into this register.

The register is reserved (must not be written) for devices that support the fully automatic trimming. Refer to Table 676: UCPD implementation.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

- **Bits 31:29** Reserved, must be kept at reset value.
- **Bits 28:25** TRIM_CC2_RP[3:0]: SW trim value for Rp current sources on the CC2 line
- **Bits 24:20** Reserved, must be kept at reset value.
- **Bits 19:16** TRIM_CC2_RD[3:0]: SW trim value for Rd resistor on the CC2 line
- **Bits 15:13** Reserved, must be kept at reset value.
- **Bits 12:9** TRIM_CC1_RP[3:0]: SW trim value for Rp current sources on the CC1 line
- **Bits 8:4** Reserved, must be kept at reset value.
- **Bits 3:0** TRIM_CC1_RD[3:0]: SW trim value for Rd resistor on the CC1 line

### 63.8.4 UCPD control register (UCPD_CR)

- **Address offset:** 0x00C
- **Reset value:** 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rs</td>
<td>rs</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:22  Reserved, must be kept at reset value.

Bit 21  **CC2TCDIS**: CC2 Type-C detector disable
The bit disables the Type-C detector on the CC2 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20  **CC1TCDIS**: CC1 Type-C detector disable
The bit disables the Type-C detector on the CC1 line.
0: Enable
1: Disable
When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19  Reserved, must be kept at reset value.

Bit 18  **RDCH**: Rdch condition drive
The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only UnattachedWait.SRC state, to respect the Type-C state. Refer to "USB Type-C ECN for Source VCONN Discharge". The CCENABLE[1:0] bitfield must be set accordingly, too.
0: No effect
1: Rdch condition drive

Bit 17  **FRSTX**: FRS Tx signaling enable.
Setting the bit enables FRS Tx signaling.
0: No effect
1: Enable
The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.1.

Bit 16  **FRSRXEN**: FRS event detection enable
Setting the bit enables FRS Rx event (FRSEVT) detection on the CC line selected through the PHYCCSEL bit. 0: Disable
1: Enable
Clear the bit when the device is attached to an FRS-incapable source/sink.

Bit 15  Reserved, must be kept at reset value.

Bit 14  Reserved, must be kept at reset value.

Bit 13  Reserved, must be kept at reset value.

Bit 12  Reserved, must be kept at reset value.

Bits 11:10  **CCENABLE[1:0]**: CC line enable
This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.
0x0: Disable both PHYs
0x1: Enable CC1 PHY
0x2: Enable CC2 PHY
0x3: Enable CC1 and CC2 PHY
A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.
Bit 9 **ANAMODE**: Analog PHY operating mode
- 0: Source
- 1: Sink
The use of CC1 and CC2 depends on CCENABLE. Refer to Table 684: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode
Refer to Table 684: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling
- 0: Use CC1 IO for Power Delivery communication
- 1: Use CC2 IO for Power Delivery communication
The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable
- 0: Disable
- 1: Enable
Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode
Determines the mode of the receiver.
- 0: Normal receive mode
- 1: BIST receive mode (BIST test data mode)
When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset
- 0: No effect
- 1: Start Tx Hard Reset message
The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet
- 0: No effect
- 1: Start Tx packet transmission
The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet
Writing the bitfield triggers the action as follows, depending on the value:
- 0x0: Transmission of Tx packet previously defined in other registers
- 0x1: Cable Reset sequence
- 0x2: BIST test sequence (BIST Carrier Mode 2)
- Others: invalid
From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).
### 63.8.5 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010  
Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Res.</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19:16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14:11</td>
<td>TYPECEVT2IE</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
</tr>
<tr>
<td>10:7</td>
<td>TYPECEVT1IE</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
</tr>
<tr>
<td>6:3</td>
<td>RXMSGENDIE</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
</tr>
<tr>
<td>2:0</td>
<td>RXORDDETIE</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
<td>...</td>
<td>Res.</td>
</tr>
</tbody>
</table>

Bits 31:21 reserved, must be kept at reset value.

- **Bit 20**: FRSEVTIE: FRSEVT interrupt enable  
  0: Disable  
  1: Enable

Bits 19:16 reserved, must be kept at reset value.

- **Bit 15**: TYPECEVT2IE: TYPECEVT2 interrupt enable  
  0: Disable  
  1: Enable

- **Bit 14**: TYPECEVT1IE: TYPECEVT1 interrupt enable  
  0: Disable  
  1: Enable

- **Bit 13**: Reserved, must be kept at reset value.

- **Bit 12**: RXMSGENDIE: RXMSGEND interrupt enable  
  0: Disable  
  1: Enable

- **Bit 11**: RXOVRIE: RXOVRIE interrupt enable  
  0: Disable  
  1: Enable

- **Bit 10**: RXHRSTDETIE: RXHRSTDET interrupt enable  
  0: Disable  
  1: Enable

- **Bit 9**: RXORDDETIE: RXORDDET interrupt enable  
  0: Disable  
  1: Enable

- **Bit 8**: RXNEIE: RXNE interrupt enable  
  0: Disable  
  1: Enable

- **Bit 7**: Reserved, must be kept at reset value.
63.8.6 UCPD status register (UCPD_SR)

Address offset: 0x014
Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

Bits 31:21 Reserved, must be kept at reset value.
Bit 20  **FRSEVT**: FRS detection event
The flag is cleared by setting the FRSEVT CF bit.
0: No new event
1: New FRS receive event occurred

Bits 19:18  **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level
The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16  **TYPEC_VSTATE_CC1[1:0]**:
The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15  **TYPECEVT2**: Type-C voltage level event on CC2 line
The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2 CF bit.
0: No new event
1: A new Type-C event

Bit 14  **TYPECEVT1**: Type-C voltage level event on CC1 line
The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2 CF bit.
0: No new event
1: A new Type-C event

Bit 13  **RXERR**: Receive message error
The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.
0: No error detected
1: Error(s) detected

Bit 12  **RXMSGEND**: Rx message received
The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.
0: No new Rx message received
1: A new Rx message received
The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11  **RXOVR**: Rx data overflow detection
The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVR CF bit.
0: No overflow
1: Overflow
The buffer overflow can occur if the received data are not read fast enough.
Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection
The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDET bit.
0: Hard Reset not received
1: Hard Reset received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detection
The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.
0: No ordered set detected
1: A new ordered set detected

Bit 8 **RXNE**: Receive data register not empty detection
The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.
0: Rx data register empty
1: Rx data register not empty

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUND**: Tx data underrun detection
The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.
0: No Tx data underrun detected
1: Tx data underrun detected

Bit 5 **HRSTSENT**: Hard Reset message sent
The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENT bit.
0: No Hard Reset message sent
1: Hard Reset message sent

Bit 4 **HRSTDISC**: Hard Reset discarded
The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.
0: No Hard Reset discarded
1: Hard Reset discarded

Bit 3 **TXMSGABT**: Transmit message abort
The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.
0: No transmit message abort
1: Transmit message abort

Bit 2 **TXMSGSSENT**: Message transmission completed
The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSSENT bit.
0: No Tx message completed
1: Tx message completed
In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.
Bit 1 **TXMSGDISC**: Message transmission discarded
The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.
- 0: No Tx message discarded
- 1: Tx message discarded
Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.

Bit 0 **TXIS**: Transmit interrupt status
The flag indicates that the UCPD_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield).
- 0: New Tx data write not required
- 1: New Tx data write required

### 63.8.7 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018
Reset value: 0x0000 0000
Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TYPECEVT2CF</td>
<td>TYPECEVT1CF</td>
<td>RXMSGENDCF</td>
<td>RXSTORCF</td>
<td>RXORDRECFCF</td>
<td>RXORDDETFCF</td>
<td>RXORDDETF</td>
<td>RXORDDET</td>
<td>TXUNDCF</td>
<td>HRSTSENCF</td>
<td>HRSTDISCCF</td>
<td>TXMSGABT</td>
<td>TXMSGSENCF</td>
<td>TXMSGDISCCF</td>
<td>Res</td>
<td>w</td>
</tr>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

- Bit 20 **FRSEVTCF**: FRS event flag (FRSEVT) clear
  Setting the bit clears the FRSEVT flag in the UCPD_SR register.

Bits 19:16 Reserved, must be kept at reset value.

- Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear
  Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register

- Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear
  Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register

- Bit 13 Reserved, must be kept at reset value.

- Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear
  Setting the bit clears the RXMSGEND flag in the UCPD_SR register.
Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear
   Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear
   Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
   Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
   Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear
   Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
   Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
   Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENCF**: Tx message send flag (TXMSGSEN) clear
   Setting the bit clears the TXMSGSEN flag in the UCPD_SR register.

Bit 1 **TXMSGDISCF**: Tx message discard flag (TXMSGDISC) clear
   Setting the bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

### 63.8.8 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

**Address offset**: 0x01C
**Reset value**: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit
   The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.
63.8.9 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020
Reset value: 0x0000 0000
Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

| Bits 31:10 | Reserved, must be kept at reset value. |
| Bits 9:0   | TXPAYSZ[9:0]: Payload size yet to transmit |
|           | The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission. |
|           | 0x2: 2 bytes - the size of Control message from the protocol layer |
|           | 0x6: 6 bytes - the shortest Data message allowed from the protocol layer |
|           | 0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer |
|           | 0x106: 262 bytes - the longest possible extended message |
|           | 0x3FF: 1024 bytes - the longest possible payload (for future expansion) |

63.8.10 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024
Reset value: 0x0000 0000
Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

| Bits 31:8  | Reserved, must be kept at reset value. |
| Bits 7:0   | TXDATA[7:0]: Data byte to transmit |
### 63.8.11 UCPD Rx ordered set register (UCPD_RX_ORDSETR)

Address offset: 0x028  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:7</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 6:4</td>
<td><strong>RXSOPKINVALID[2:0]</strong>:</td>
</tr>
<tr>
<td></td>
<td>The bitfield is for debug purposes only.</td>
</tr>
<tr>
<td></td>
<td>0x0: No K-code corrupted</td>
</tr>
<tr>
<td></td>
<td>0x1: First K-code corrupted</td>
</tr>
<tr>
<td></td>
<td>0x2: Second K-code corrupted</td>
</tr>
<tr>
<td></td>
<td>0x3: Third K-code corrupted</td>
</tr>
<tr>
<td></td>
<td>0x4: Fourth K-code corrupted</td>
</tr>
<tr>
<td></td>
<td>Others: Invalid</td>
</tr>
<tr>
<td>Bit 3</td>
<td><strong>RXSOP3OF4</strong>:</td>
</tr>
<tr>
<td></td>
<td>The bit indicates the number of correct K-codes. For debug purposes only.</td>
</tr>
<tr>
<td></td>
<td>0: 4 correct K-codes out of 4</td>
</tr>
<tr>
<td></td>
<td>1: 3 correct K-codes out of 4</td>
</tr>
<tr>
<td>Bits 2:0</td>
<td><strong>RXORDSET[2:0]</strong>: Rx ordered set code detected</td>
</tr>
<tr>
<td></td>
<td>0x0: SOP code detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x1: SOP' code detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x2: SOP'' code detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x3: SOP'_Debug detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x4: SOP''_Debug detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x5: Cable Reset detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x6: SOP extension#1 detected in receiver</td>
</tr>
<tr>
<td></td>
<td>0x7: SOP extension#2 detected in receiver</td>
</tr>
</tbody>
</table>
63.8.12 UCPD Rx payload size register (UCPD_RX_PAYSZR)

Address offset: 0x02C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 9:0 **RXPAYSZ[9:0]**: Rx payload size received

This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).

- 0x2: 2 bytes - the size of Control message from the protocol layer
- 0x6: 6 bytes - the shortest Data message allowed from the protocol layer
- 0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
- 0x106: 262 bytes - the longest possible extended message
- 0x3FF: 1024 bytes - the longest possible payload (for future expansion)

The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

63.8.13 UCPD receive data register (UCPD_RXDR)

Address offset: 0x030
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **RXDATA[7:0]**: Data byte received
63.8.14 UCPD Rx ordered set extension register 1
(UCPD_RX_ORDEXT1)

Address offset: 0x034
Reset value: 0x0000 0000
Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

Bits 31:20 Reserved, must be kept at reset value.
Bits 19:0 RXSOPX1[19:0]: Ordered set 1 received
The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

63.8.15 UCPD Rx ordered set extension register 2
(UCPD_RX_ORDEXT2)

Address offset: 0x038
Reset value: 0x0000 0000
Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

Bits 31:20 Reserved, must be kept at reset value.
Bits 19:0 RXSOPX2[19:0]: Ordered set 2 received
The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.
## 63.8.16 UCPD register map

### Table 688. UCPD register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | UCPD_CFRG1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |               | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | UCPD_CFRG2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | UCPD_CFRG3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x00C  | UCPD_CR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x010  | UCPD_IMR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x014  | UCPD_SR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Table 688. UCPD register map and reset values (continued)

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x018  | UCPD_ICR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x01C  | UCPD_TX_ORDSETR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x020  | UCPD_TX_PAYSZR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x024  | UCPD_TXDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x028  | UCPD_RX_ORDSETR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x02C  | UCPD_RX_PAYSZR |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x030  | UCPD_RXDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x034  | UCPD_RX_ORDEXTR1 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x038  | UCPD_RX_ORDEXTR2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x03C - 0x3FF | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Refer to Section 2.3 on page 149 for the register boundary addresses.
64 Ethernet (ETH): media access control (MAC) with DMA controller

64.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2008 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications.

64.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

64.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:

- IEEE 802.3-2008 for Ethernet MAC and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- AMBA 2.0 for AHB master and AHB slave ports
- RMII specification version 1.2 from RMII consortium

64.2.2 MAC features

MAC Tx and Rx common features

- Separate transmission, reception, and control interfaces to the application
- 10, 100 Mbps data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RMII interface to communicate with an external Fast Ethernet PHY
- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using backpressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII PHYs
• 32-bit data transfer interface on the application side
• Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
• Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
• Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
• Flexibility to control pulse-per-second (PPS) output signal (eth_ptp_pps_out and ETH_PPS_OUT)
• MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

MAC Tx features
• Preamble and start-of-frame data (SFD) insertion
• Separate 32-bit status for each packet transmitted from the application
• Automatic CRC and pad generation controllable on a per-frame basis
• Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
• Programmable Inter Packet Gap (40–96 bit times in steps of 8)
• IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
• Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
• Insertion, replacement, or deletion of up to two VLAN tags
• Option to transmit packets with reduced preamble size in Full-duplex mode
• Insert, replace, or delete queue/channel-based VLAN tags

MAC Rx features
• Automatic Pad and CRC stripping options
• Option to disable automatic CRC checking
• Preamble and SFD deletion
• Separate 112-bit or 128-bit status
• Programmable watchdog timeout limit
• Flexible address filtering modes:
  – Four 48-bit perfect (DA) address filters with masks for each byte
  – Four 48-bit SA address comparison check with masks for each byte
  – 64 bit Hash filter for multicast and unicast (DA) addresses
• Option to pass all multicast addressed packets
• Promiscuous mode to pass all packets without any filtering for network monitoring
• Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Detection of remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in Full-duplex mode)
- Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN tags and providing the tags in the status

64.2.3 Transaction layer (MTL) features

MTL Tx and Rx Common Features
- 32-bit Transaction Layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features
- 2048-byte Transmit FIFO with programmable threshold capability
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion
  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
MTL Rx features

- 2048-byte Receive FIFO with configurable threshold
- Programmable Rx queue threshold (default fixed at 64 bytes) in Threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level

64.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see Section 64.11.2: Ethernet DMA registers) to control the DMA operations. The DMA block supports the following features:

- 32-bit data transfers
- Separate DMA in Transmit path and receive paths
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors
- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet Transmit or Receive Complete Interrupt control
- Round-robin or fixed-priority arbitration between the Receive and Transmit engines
- Start and Stop modes
- Separate ports for host control (AHB) access and host data interface
- Tx DMA channel with TCP segmentation offload (TSO) feature enabled
- Programmable control for Transmit Descriptor posted writes to improve the throughput

64.2.5 Bus interface features

AHB master interface

The AHB master interface features are the following:

- Interfaces with the application through AHB
- 32-bit data on the AHB master port
- Split, Retry, and Error AHB responses
- AHB 1-Kbyte boundary burst splitting
- Software-selected type of AHB burst (fixed burst, indefinite burst, or mix of both)
The AHB master interface does not generate the following:
- Wrap burst
- Locked or protected transfers

**AHB slave interface**

The AHB slave interface supports the following features:
- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:
- Split
- Retry
- Error

### 64.3 Ethernet pins and internal signals

*Table 689* lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII or RMII) and on the device configuration. *Table 690* shows the internal Ethernet signals.

<table>
<thead>
<tr>
<th>Port name</th>
<th>Digital port type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH_COL</td>
<td>Input</td>
<td>Collision detection signal, MII only.</td>
</tr>
<tr>
<td>ETH_CRS</td>
<td>input</td>
<td>Carrier sense signal, MII only</td>
</tr>
<tr>
<td>ETH_REF_CLK</td>
<td>Input</td>
<td>RMII reference clock</td>
</tr>
<tr>
<td>ETH_RX_CLK</td>
<td>Input</td>
<td>MII timing reference for Rx data transfers</td>
</tr>
<tr>
<td>ETH_RXD[3:0]</td>
<td>Input</td>
<td>Receive data. 4 pins for MII, 2 for RMII.</td>
</tr>
<tr>
<td>ETH_RX_DV</td>
<td>Input</td>
<td>Receive data valid</td>
</tr>
<tr>
<td>ETH_CRS_DV</td>
<td>Input</td>
<td>RMII: Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.</td>
</tr>
<tr>
<td>ETH_RX_ER</td>
<td>Input</td>
<td>Receive error</td>
</tr>
<tr>
<td>ETH_TX_CLK</td>
<td>Input</td>
<td>MII timing reference for Tx data transfers</td>
</tr>
<tr>
<td>ETH_TXD[3:0]</td>
<td>Output</td>
<td>Transmit data. 4 pins for MII, 2 for RMII.</td>
</tr>
<tr>
<td>ETH_TX_EN</td>
<td>Output</td>
<td>Transmit data enable</td>
</tr>
<tr>
<td>ETH_TX_ER</td>
<td>Output</td>
<td>Transmit error</td>
</tr>
<tr>
<td>ETH_MDC</td>
<td>Output</td>
<td>Management data clock</td>
</tr>
<tr>
<td>ETH_MDIO</td>
<td>Input/output</td>
<td>Management data</td>
</tr>
</tbody>
</table>
The Ethernet peripheral is composed of 4 main functional modules:

- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)
  This is the logical DMA module with one physical channel for reception and one for transmission. It controls the data transfers between MAC and system memory through the AMBA AHB 32-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII PHY Media Independent Interfaces.
64.4.1 DMA controller

The DMA has independent Transmit (Tx) and Receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet Transmit and Receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.
The base address of each list is written to the respective Tx and Rx registers: Channel Tx descriptor list address register (ETH_DMACTXDLAR) and Channel Rx descriptor list address register (ETH_DMACRXDLAR).

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, Channel Tx descriptor ring length register (ETH_DMACTXRLR) and Channel Rx descriptor ring length register (ETH_DMACRXRLR).

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

Descriptors are specified in Section 64.10: Descriptors.

**DMA arbitration**

The DMA module incorporates an arbiter that performs the arbitration between the Tx and Rx channels accesses from the AHB master interface. The following two types of arbitrations are supported and can be selected through DMA mode register (ETH_DMAMR):

- **Round-robin arbitration**: the arbiter allocates the data bus between Rx and Tx in ratio set by Bits [14:12] of ETH_DMAMR.
- **Fixed-priority arbitration**: by default Rx DMA always gets priority over Tx DMA for data access. Setting bit 11 of ETH_DMAMR register gives priority to the Tx DMA.
DMA transmission in default mode

The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the descriptor tail pointer offset value of the Transmit channel.
3. The DMA fetches the descriptor from the application memory.
4. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   - The descriptor is flagged as owned by the application (TDES3[31] = 0).
   - The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   - An error condition occurs.
5. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
6. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
7. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
8. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.
9. Bit 0 of Channel status register (ETH_DMACSR) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
10. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR) when it receives a Transmit Poll demand and the Underflow Interrupt Status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.
DMA transmission in OSP (Operate on Second Packet) mode

In Run state, if bit 4 is set in the Channel transmit control register (ETH_DMACTXCR), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see Section: DMA transmission in default mode).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.
   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in Transmit Descriptor Tail Pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in Figure 954: Receive DMA flow.
Figure 953. DMA transmission flow (OSP mode)

1. Start Tx DMA
2. (Re-)fetch next descriptor from Tx Queue
3. Own bit set?
   - Yes: Transfer data from Buffer(s)
   - No: Wait for previous packet status
4. Packet read completely?
   - Yes: Second packet?
   - No: Wait for previous packet status
5. Second packet?
   - Yes: Write timestamp to TDES0 and TDES1
   - No: Close intermediate descriptor
6. Close intermediate descriptor
7. Timestamp present?
   - Yes: Wait status word to TDES3
   - No: No
8. End of descriptor ring?
   - Yes: Tx DMA queue suspended
   - No: Prev. packet status available?
9. Prev. packet status available?
   - Yes: Tx Poll demand?
   - No: DMA Tx stopped?
10. DMA Tx stopped?
    - Yes: Stop DMA
    - No: Yes

MS40863V1
DMA reception

In the Receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

The DMA Rx descriptor ring structure is described in **Section 64.10: Descriptors**.

The reception sequence for Rx DMA engine is as follows (see also **Figure 954: Receive DMA flow**):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application should set the correct value in the Receive descriptor tail pointer register of corresponding DMA channel to indicate the location of the last valid descriptor for the DMA. If the tail pointer points to descriptor N, the last valid descriptor for the DMA is descriptor N - 1.

2. When bit 0 of **Channel receive control register (ETH_DMACRXCR)** is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. The descriptors referenced between the current descriptor and the tail pointer registers are available for the DMA. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.

3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.

4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).

5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.

6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.

7. The DMA retrieves the status of the Receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.

8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.

9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).

10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).

11. The Receive DMA exits the Suspend state when a Receive Poll demand is given, and the application updates the channel Receive descriptor tail pointer register to indicate the location of the last valid descriptor for DMA. Then, the engine proceeds to step 2 and fetches again the next descriptor.

**Note:** Refer to **Section : Descriptor tail pointer handling examples** for updating the correct value in receive descriptor tail pointer register.
**Figure 954. Receive DMA flow**

- **Start Rx DMA**
- **(R)eFetch next descriptor**
- **Own bit set?**
  - Yes: **(R)eFetch next descriptor**
  - No: **Own bit set?**
- **Packet data available?**
  - Yes: **Write data to buffer**
  - No: **Wait for packet data**
- **Packet transfer complete?**
  - Yes: **Set pending timestamp**
  - No: **Close RDES3 as intermediate descriptor**
- **Rx DMA stopped?**
  - Yes: **Clear pending timestamp**
  - No: **End of descriptor ring?**
    - Yes: **Write timestamp to RDES0 and RDES1**
    - No: **(R)eFetch next descriptor**
- **Timestamp pending?**
  - Yes: **Write timestamp to RDES0 and RDES1**
  - No: **Clear pending timestamp**
- **Receive Poll demand**
  - Yes: **Start Rx DMA**
  - No: **Supsend Rx DMA**
Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers. The DMA arbiter supports two types of arbitration: fixed priority and weighted round-robin. The DA bit of the DMA mode register (ETH_DMAMR) specifies the arbitration scheme (fixed or weighted round-robin) between the Tx and Rx DMA of a given channel.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the DMA mode register (ETH_DMAMR). For round-robin arbitration, the weighted priority between the Tx DMA and Rx DMA is configured through the PR field of the DMA mode register (ETH_DMAMR). Table 691 provides information about the priority scheme between Tx DMA and Rx DMA.

Table 691. Priority scheme for Tx DMA and Rx DMA

<table>
<thead>
<tr>
<th>DMA mode register (ETH_DMAMR)</th>
<th>Priority scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR[2:0] TXPR DA</td>
<td></td>
</tr>
<tr>
<td>x x x 0 1</td>
<td>Rx always has priority over Tx</td>
</tr>
<tr>
<td>0 0 0 0 0</td>
<td>Tx and Rx have equal priority. Rx gets the access first on simultaneous requests.</td>
</tr>
<tr>
<td>0 0 1 0 0</td>
<td>Rx has priority over Tx in ratio 2:1.</td>
</tr>
<tr>
<td>0 1 0 0 0</td>
<td>Rx has priority over Tx in ratio 3:1.</td>
</tr>
<tr>
<td>0 1 1 0 0</td>
<td>Rx has priority over Tx in ratio 4:1.</td>
</tr>
<tr>
<td>1 0 0 0 0</td>
<td>Rx has priority over Tx in ratio 5:1.</td>
</tr>
<tr>
<td>1 1 0 0 0</td>
<td>Rx has priority over Tx in ratio 6:1.</td>
</tr>
<tr>
<td>1 1 1 0 0</td>
<td>Rx has priority over Tx in ratio 7:1.</td>
</tr>
<tr>
<td>1 x x x 1 1</td>
<td>Tx always has priority over Rx.</td>
</tr>
<tr>
<td>0 0 0 1 0</td>
<td>Tx and Rx have equal priority. Tx gets the access first on simultaneous requests.</td>
</tr>
<tr>
<td>0 0 1 1 0</td>
<td>Tx has priority over Rx in ratio 2:1.</td>
</tr>
<tr>
<td>0 1 0 1 0</td>
<td>Tx has priority over Rx in ratio 3:1.</td>
</tr>
<tr>
<td>0 1 1 1 0</td>
<td>Tx has priority over Rx in ratio 4:1.</td>
</tr>
<tr>
<td>1 0 0 1 0</td>
<td>Tx has priority over Rx in ratio 5:1.</td>
</tr>
<tr>
<td>1 0 1 1 0</td>
<td>Tx has priority over Rx in ratio 6:1.</td>
</tr>
<tr>
<td>1 1 0 1 0</td>
<td>Tx has priority over Rx in ratio 7:1.</td>
</tr>
<tr>
<td>1 1 1 1 0</td>
<td>Tx has priority over Rx in ratio 8:1.</td>
</tr>
</tbody>
</table>
64.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 32-bit wide data paths: the Transmit path and the Receive Path.

- **Transmit path**
  The application or internal DMA pushes the Ethernet packets read from the application or system memory into the Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 2048 bytes.

- **Receive path**
  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured Receive threshold (RTC bits[1:0] defined in *Rx queue operating mode register (ETH_MTLRXQOMR)*), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 2048 bytes.

64.4.3 MAC

The MAC is responsible of the Ethernet protocol processing. In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

**MAC transmission**

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:
     - the packet was successfully transmitted;
     - the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.
- If any one of the following event happens, the MAC aborts the packet transmission:
  - no carrier (Half-duplex mode)
  - loss of carrier (Half-duplex mode)
  - excessive deferral (Half-duplex mode)
  - late collisions (Half-duplex mode)
  - jabber
  the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.

5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

*Figure 955: Overview of MAC transmission flow* illustrates the MAC transmission process flow.
Figure 955. Overview of MAC transmission flow

Start

Wait for data & SOP From MTL

SOP asserted by MTL?

YES

Wait for IPG/any back-off delay (half-duplex)

Transmit preamble+ SFD+Data received from the MTL to the PHY

No carrier/Carrier loss/Excessive deferral/Late collisions/Jabber?

YES

Drop all the data received from MTL and abort transmission

NO

Collision

YES

Send status to MTL with Retry bit set

Condition B

Condition D: Retry_count ≤ retry_limit

Condition A

Drop all the data received from MTL and abort transmission

NO

A. EOP asserted by MTL?
B. Underflow asserted by MAC?

Condition B

Normal transmission completed and Transmission status conveyed to MTL
MAC reception

A receive operation is initiated when the MAC detects an SFD on MII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of MII becomes active, the Receive State Machine (RSM) starts looking for the SFD field (0xD nibble).
   
   The state machine drops received packets until it detects SFD.

2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).

3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application. The MAC converts the received nibble data into bytes and forwards the valid packet data to the RFC module.

4. The receive state machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the Operating mode configuration register (ETH_MACCR)), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.

5. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the Operating mode configuration register (ETH_MACCR). However, if the CRC stripping has been enabled for Type packets and not enabled the Receive Checksum Offload Engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.

6. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of Watchdog timeout register (ETH_MACWTR)) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of Operating mode configuration register (ETH_MACCR). However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.
Figure 956. MAC reception flow

1. Start
2. Wait for phy_rxdv_i from GMII/MII
3. Wait until SFD is detected
4. Start sending the received Ethernet packet to the application layer, starting from DA field
5. IEEE 1588 timestamp feature enabled?
   - Yes: Store the snapshot of the system time when SFD detected
   - No: Length/Type field ≤1536
     - Length/Type field ≥1536
       - CRC stripping for Type packets enabled?
         - Yes: Receive checksum Offload engine enabled?
           - Yes: Send all received Ethernet packet data to RFC
           - No: Drop the last 4 bytes of all Ether type packets and send the remaining bytes to RFC
         - No: CRC error? OR Packet to be filtered?
           - No: Send the packet, timestamp and Status to the Application layer
           - Yes: Drop the packet and send the Status to the Application layer
       - NO: Send only the number of bytes (specified in the Length field) of the received Ethernet packet to RFC. Drop extra padding and FCS field
64.5 Ethernet functional description: MAC

64.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:
- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the Receive path. Stripping and providing up to two VLAN Tags in the Receive path as a part of the Receive status

Transmit path

*Table 692: Double VLAN processing features in Tx path* describes the features supported by the MAC on the Transmit side.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
</table>
| Support for C-VLAN and S-VLAN Tag types | The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following:
  - The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled.
  - The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled.
  - The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN.
  - The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN. |
| VLAN Tag deletion | VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag. |
| VLAN Tag Insertion or Replacement | VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word. |
Table 693: Double VLAN processing in Rx path describes the features supported by the MAC on the Receive side and the corresponding bits in the VLAN tag register (ETH_MACVTR).

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outer or inner VLAN tag-based filtering</td>
<td>The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.</td>
</tr>
<tr>
<td>C-VLAN or S-VLAN tag-based filtering</td>
<td>The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.</td>
</tr>
<tr>
<td>Outer and Inner VLAN Tag stripping</td>
<td>The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.</td>
</tr>
<tr>
<td>16-bit outer and inner VLAN Tag and Type in Rx status</td>
<td>The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.</td>
</tr>
<tr>
<td>Disabling or skipping checking of outer VLAN Tag type</td>
<td>The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.</td>
</tr>
</tbody>
</table>

### 64.5.2 Source address and VLAN insertion, replacement, or deletion

#### Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets
  
  To enable this feature for all packets, program the SARC field of the Operating mode configuration register (ETH_MACCR).

- Enabling SA insertion or replacement for selective packets
  
  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of Transmit Descriptor Word 3/TDES3, refer to Section 64.10.3: Transmit descriptor) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or
replacement by MAC Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

**VLAN insertion, replacement, or deletion**

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

Insertion or replacement is performed based on the setting of VLTI bit in the VLAN inclusion register (ETH_MACVIR) as described in **Table 694: VLAN insertion or replacement based on VLTI bit**.

**Table 694. VLAN insertion or replacement based on VLTI bit**

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLTI bit is set</td>
<td>The MAC inserts or replaces the following:</td>
</tr>
<tr>
<td></td>
<td>VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of VLAN inclusion register (ETH_MACVIR))</td>
</tr>
<tr>
<td></td>
<td>VLAN Tag field with VT field of Transmit context descriptor of the packet</td>
</tr>
<tr>
<td>VLTI bit is reset</td>
<td>The MAC inserts or replaces the following:</td>
</tr>
<tr>
<td></td>
<td>VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of VLAN inclusion register (ETH_MACVIR))</td>
</tr>
<tr>
<td></td>
<td>VLAN Tag field with the VLT field of VLAN inclusion register (ETH_MACVIR)</td>
</tr>
</tbody>
</table>

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of VLAN inclusion register (ETH_MACVIR).
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see **Table 727: TDES2 normal descriptor (read format)**).

In addition, the VLP (VLAN Priority control) bit must be reset in VLAN inclusion register (ETH_MACVIR) (for outer VLAN) and Inner VLAN inclusion register (ETH_MACIVIR) (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.
64.5.3 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- **MAC source or destination address filtering**: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- **VLAN filtering**: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- **Layer 3 and Layer 4 filtering**: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 957* shows the filtering sequence for Rx packets.

*Figure 957. Packet filtering sequence*

The sequence shown in *Figure 957* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the
specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of Packet filtering control register (ETH_MACPFR) control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

**MAC source or destination address filtering**

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

**Unicast destination address filtering**

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of Packet filtering control register (ETH_MACPFR) is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in MAC Address x high register (ETH_MACAxHR). This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

**Multicast destination address filtering**

To program the MAC to pass all multicast packets, set the PM bit in Packet filtering control register (ETH_MACPFR). If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the Packet filtering control register (ETH_MACPFR).

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

**Hash or Perfect address filtering**

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in Packet filtering control register (ETH_MACPFR). This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

**Broadcast address filtering**

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in Packet filtering control register (ETH_MACPFR).

**Unicast source address filtering**
The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of MAC Address x high register (ETH_MACAxHR).

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in Packet filtering control register (ETH_MACPFR). Otherwise, the result of the SA filter is given as a status bit in the Receive Status word (see Table 696). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

**Inverse filtering**

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of Packet filtering control register (ETH_MACPFR). The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 695 and Table 696 summarize the DA and SA filtering based on the type of packets received.

**Note:** When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

### Table 695. Destination address filtering

<table>
<thead>
<tr>
<th>Packet type</th>
<th>PR</th>
<th>HPF</th>
<th>HUC</th>
<th>DAIF</th>
<th>HMC</th>
<th>PM</th>
<th>DBF</th>
<th>DA filter operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broadcast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass all packets</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td>Unicast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fail on Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Hash filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fail on Hash filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass on Hash or Perfect/Group filter match</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Fail on Hash or Perfect/Group filter match</td>
</tr>
<tr>
<td>Multicast</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Pass all packets</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>Pass all packets</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Pass on Hash filter match and drop Pause packets if PCF = 0x</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x</td>
</tr>
</tbody>
</table>
The MAC supports Perfect and Hash VLAN filtering. Refer to Section 64.9.14: Programming guidelines to perform VLAN filtering on the receiver for detailed programming steps.

**VLAN tag Perfect filtering**

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of VLAN tag register \( \text{ETH\_MACVTR} \). In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of VLAN tag register \( \text{ETH\_MACVTR} \). The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

**Note:** The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

**VLAN tag Hash filtering**

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the VLAN tag register \( \text{ETH\_MACVTR} \). If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.
Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag register (ETH_MACVTR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 697 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are received and the VLAN match status is indicated in the VF bit of RDES2 normal descriptor (write-back format). When the RA bit is not set and the VTFE bit is set in Packet filtering control register (ETH_MACPFR), the packet is dropped if the final VLAN match status is Fail. In Table 697, value X means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of VLAN tag register (ETH_MACVTR), all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in VLAN tag register (ETH_MACVTR).

<table>
<thead>
<tr>
<th>VID</th>
<th>VLAN perfect filter match result</th>
<th>VTHM Bit</th>
<th>VLAN Hash filter match result</th>
<th>VTIM bit</th>
<th>Final VLAN match status</th>
</tr>
</thead>
<tbody>
<tr>
<td>VID = 0</td>
<td>Pass</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>1</td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>1</td>
<td>Pass</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td>VID ≠ 0</td>
<td>Pass</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Fail</td>
<td>0</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
<td>0</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>Pass</td>
</tr>
<tr>
<td></td>
<td>Pass</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
<td>1</td>
<td>Fail</td>
</tr>
<tr>
<td></td>
<td>Fail</td>
<td>1</td>
<td>Fail</td>
<td>1</td>
<td>Pass</td>
</tr>
</tbody>
</table>

1. In this table, ‘X’ represents any value.
Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the Receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the Operating mode configuration register (ETH_MACCR) to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets
  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of Operating mode configuration register (ETH_MACCR) is set and one of the following conditions is true:
    - All enabled Layer 3 and Layer 4 fields match.
    - At least one of the enabled field matches and other fields are bypassed or disabled
      When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

  Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets
  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets. The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value, such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

- Non-TCP or UDP IP Packets
  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.
For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as L3 and L4 control 0 register (ETH_MACL3L4C0R), to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- Layer4 Address filter 0 register (ETH_MACL4A0R)
- Layer3 Address 0 filter 0 register (ETH_MACL3A00R)
- Layer3 Address 1 filter 0 register (ETH_MACL3A10R)
- Layer3 Address 2 filter 0 register (ETH_MACL3A20R)
- Layer3 Address 3 filter 0 register (ETH_MACL3A30R)

The second, and independent set of registers are: L3 and L4 control 1 register (ETH_MACL3L4C1R), Layer 4 address filter 1 register (ETH_MACL4A1R), Layer3 address 0 filter 1 Register (ETH_MACL3A01R), Layer3 address 1 filter 1 register (ETH_MACL3A11R), Layer3 address 2 filter 1 Register (ETH_MACL3A21R) and Layer3 address 3 filter 1 register (ETH_MACL3A31R).

64.5.4 IEEE 1588 timestamp support

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- IEEE 1588 timestamp support
- IEEE 1588 system time source
- IEEE 1588 auxiliary snapshots
- Flexible pulse-per-second output
IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588-2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock
  
The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.
  
The ordinary clock can be a grandmaster or a slave clock. It supports the following features:
  
  - Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in Timestamp control Register (ETH_MACTSCR).
  - Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

  **Table 698. Ordinary clock: PTP messages for snapshot**

<table>
<thead>
<tr>
<th>Master</th>
<th>Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay_Req</td>
<td>SYNC</td>
</tr>
</tbody>
</table>

For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in Timestamp control Register (ETH_MACTSCR).

- Boundary clock
The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:
- The clock data sets are common to all ports of the boundary clock.
- The local clock is common to all ports of the boundary clock.

- **End-to-end transparent clock**
  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the Ingress port to the Egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both Ingress and Egress ports only for the messages mentioned in Table 699. You can take the snapshot by setting the SNAPTYPESEL bits to 10 in the Timestamp control Register (ETH_MACTSCCR).

<table>
<thead>
<tr>
<th>PTP messages</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNC</td>
<td></td>
</tr>
<tr>
<td>Delay_Req</td>
<td></td>
</tr>
</tbody>
</table>

- **Peer-to-peer transparent clock**
  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdelay_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in Table 707.
You can take the snapshot by setting the SNAPTYPESEL bit to 11 in Timestamp control Register (ETH_MACTSCR).

**Delay request-response mechanism**

The system or network is classified into the master and slave nodes for distributing the timing and clock information. Figure 958 shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

**Figure 958. Networked time synchronization**

As shown in Figure 958, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at $t_1$. This time must be captured for Ethernet ports at MII.
2. The slave receives the SYNC message and also captures the exact time, $t_2$, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains $t_1$ information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, $t_3$, at which this packet leaves the MII interface.
5. The master receives the message, capturing the exact time \( t_4 \), at which the message enters its system.

6. The master sends the \( t_4 \) information to the slave in the Delay_Resp message.

7. The slave uses the four values of \( t_1, t_2, t_3, \) and \( t_4 \) to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

**Peer-to-peer PTP transparent clock (P2P TC) message support**

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow_up, and Delay Response messages. *Figure 959* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

**Figure 959. Propagation delay calculation in clocks supporting peer-to-peer path correction**

As shown in *Figure 959*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (\( t_1 \)) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (\( t_2 \)) for this message.
3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t₃) for this message.
   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:
   – Difference between the timestamps t₂ and t₃ in the Pdelay_Resp message
   – Difference between the timestamps t₂ and t₃ in the Pdelay_Resp_Follow_Up message
   – Timestamps t₂ and t₃ in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively

4. Port 1 generates a timestamp (t₄) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

**Timestamp correction**

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (MII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the MII and PTP clocks.

**Ingress correction**

In the Receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the Timestamp Ingress correction nanosecond register (ETH_MACTSICNR).

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core
   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the input of the core to the internal capture point
   The latency differs based on the active PHY interface (such as MII or RMII) and the operating speed, as shown in Table 701.

3. CDC synchronization
   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).
The values determined from these three components should be added by the software and must be written into the TSIC field of the *Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)*.

**Note:** The value written to the register must be negative (two’s complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with "10^9 – total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is −5 ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)* is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of −0.466 ns. Therefore, bits[30:0] must be written with "2^{31} – total ingress_correction_value" represented in binary with bit[31] = 1.

**Egress correction**

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp Egress correction nanosecond register (ETH_MACTSECNR)*.

The correction value consists of the following three components:

1. **External latency in the PHY layer between the output of the core and the boundary of the port and the network**
   
   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.

2. **Internal latency from the internal capture point and the output of the core**
   
   The latency differs based on the active PHY interface (RMII, MII, etc.) and the operating speed as shown in *Table 701*.

3. **CDC synchronization error**
   
   The timestamp correction because of synchronization is compensated by adding
   
   \[
   \text{EGRESS}_\text{SYNC}_\text{CORR} = (1 \times \text{PTP}_\text{CLK}_\text{PER} + 4 \times \text{TX}_\text{CLK}_\text{PER})
   \]
   
   *Table 701* lists the Egress and Ingress latency values for various PHY interfaces:

<table>
<thead>
<tr>
<th>PHY interface</th>
<th>Egress latency</th>
<th>Ingress latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGMII 1 Gbps</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>RGMII 100 Mbps</td>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>RGMII 10 Mbps</td>
<td>400</td>
<td>400</td>
</tr>
</tbody>
</table>
Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- **Maximum PTP clock frequency**
  
  The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- **Minimum PTP clock frequency**
  
  The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the MII clock domain. This relationship is given by the following equation:

  \[
  3 \times \text{PTP clock period} + 4 \times \text{MII clock period} \leq \text{Minimum gap between two SFDs}
  \]

  The MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in Table 702.

### Table 702. Minimum PTP clock frequency example

<table>
<thead>
<tr>
<th>Mode</th>
<th>Minimum gap between two SFDs</th>
<th>Minimum PTP frequency with internal timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 Mbps Full-duplex</td>
<td>168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)</td>
<td>5 MHz</td>
</tr>
<tr>
<td>10 Mbps Half-duplex</td>
<td>48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)</td>
<td>5 MHz</td>
</tr>
<tr>
<td>100 Mbps full duplex</td>
<td>168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)</td>
<td>5 MHz</td>
</tr>
<tr>
<td>100 Mbps Half-duplex</td>
<td>48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)</td>
<td>5 MHz</td>
</tr>
</tbody>
</table>

PTP processing and control
Table 703 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

### Table 703. Message format defined in IEEE 1588-2008

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octet</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transportSpecific</td>
<td>messageType</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td>versionPTP</td>
<td>1</td>
</tr>
<tr>
<td>messageLength</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>domainNumber</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>flagField</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>correctionField</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Reserved</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>sourcePortIdentity</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>sequenceId</td>
<td>2</td>
<td>30</td>
</tr>
<tr>
<td>controlField(1)</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>logMessageInterval</td>
<td>1</td>
<td>33</td>
</tr>
</tbody>
</table>

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

#### PTP packets over IPv4

Table 704 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in Table 703.

### Table 704. Message format defined in IEEE 1588-2008

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC packet type</td>
<td>12, 13</td>
<td>0x0800</td>
<td>IPv4 datagram</td>
</tr>
<tr>
<td>IP version and header</td>
<td>14</td>
<td>0x45</td>
<td>IP version is IPv4</td>
</tr>
<tr>
<td>length</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layer 4 protocol</td>
<td>23</td>
<td>0x11</td>
<td>UDP</td>
</tr>
</tbody>
</table>
PTP frames over IPv6

Table 705 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in Table 703.

Table 705. IPv6-UDP PTP packet fields required for control and status

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC packet type</td>
<td>12, 13</td>
<td>0x86DD</td>
<td>IP datagram</td>
</tr>
<tr>
<td>IP version</td>
<td>14 (bits [7:4])</td>
<td>0x6</td>
<td>IP version is IPv6</td>
</tr>
<tr>
<td>Layer 4 protocol</td>
<td>20(1)</td>
<td>0x11</td>
<td>UDP</td>
</tr>
</tbody>
</table>
PTP packets over Ethernet

Table 706 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 706. Ethernet PTP packet fields required for control and status

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC destination multicast address(^{(1)})</td>
<td>0–5</td>
<td>01-B-19-00-00-00 01-80-C2-00-00-0E</td>
<td>All PTP messages can use any of the following multicast addresses(^{(2)}): 01-B-19-00-00-00 01-80-C2-00-00-0E(^{(3)})</td>
</tr>
<tr>
<td>MAC packet type</td>
<td>12, 13</td>
<td>0x88F7</td>
<td>PTP Ethernet packet</td>
</tr>
</tbody>
</table>
Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the Transmit Descriptor (see Section 64.10.3: Transmit descriptor). The MAC returns the timestamp to the software inside the corresponding Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the Timestamp control Register (ETH_MACTSCR):

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp

Table 706. Ethernet PTP packet fields required for control and status (continued)

<table>
<thead>
<tr>
<th>Matched field</th>
<th>Octet position</th>
<th>Matched value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PTP control field (IEEE 1588 version 1)</td>
<td>46</td>
<td>0x00, 0x01, 0x02, 0x03, or 0x04</td>
<td>0x00: SYNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x01: Delay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x02: Follow_Up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x03: Delay_Cmd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x04: Management</td>
</tr>
<tr>
<td>PTP message type field (IEEE 1588 version 2)</td>
<td>14 (nibble)</td>
<td>0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD</td>
<td>0x0: SNAP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x1: Delay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x2: PDelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x3: PDelay_Cmd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x8: Follow_Up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x9: Delay_Cmd</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xA: PDelay_Cmd Follow_Up</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xB: Announce</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xC: Signaling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0xD: Management</td>
</tr>
<tr>
<td>PTP version</td>
<td>15 (nibble)</td>
<td>0x1 or 0x2</td>
<td>0x1: Supports PTP version 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x2: Supports PTP version 2</td>
</tr>
</tbody>
</table>

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the **TSENMACADDR** bit of Timestamp control Register (ETH_MACTSCR) is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.
Ethernet (ETH): media access control (MAC) with DMA controller

- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type
  This feature controls the type of messages for which snapshots are taken.

**Note:** The peripheral also supports the PTP messages over VLAN packets.

*Table 707* indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPESEL field in *Timestamp control Register (ETH_MACTSCR)*.

<table>
<thead>
<tr>
<th>SNAPTYPESEL</th>
<th>TSMSTRENA</th>
<th>TSEVNTENA</th>
<th>PTP messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>X</td>
<td>0</td>
<td>SYNC, Follow_Up, Delay_Req, Delay_Resp</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>SYNC</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Delay_Req</td>
</tr>
<tr>
<td>01</td>
<td>X</td>
<td>0</td>
<td>SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>SYNC, Pdelay_Req, Pdelay_Resp</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>Delay_Req, Pdelay_Req, Pdelay_Resp</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>X</td>
<td>SYNC, Delay_Req</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>Pdelay_Req, Pdelay_Resp</td>
</tr>
</tbody>
</table>

The DMA returns the timestamp to the software application inside the corresponding Receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

**Programming guidelines for IEEE 1588 timestamping (system time correction)**

See *Section : System time correction* in *Section 64.9.9: Programming guidelines for IEEE 1588 timestamping on page 3366*.

**IEEE 1588 system time source**

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

**Description of IEEE 1588 system time source**

The peripheral uses the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:
- Ulinteger48 seconds field
The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as secondsField = 0x0000 0000 0002.

- **UInteger32 nanosecondsField**
  
The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.

  The nanoseconds field supports the following two modes:
  - **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.
  - **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is \(~0.466\) ns per bit.

  These modes can be set through TSCTRLSSR bit in *Timestamp control Register (ETH_MACTSCR)*.

**System time register module**

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the MII.

The system time counter can be initialized or corrected using either the coarse or the fine correction method.

In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 960*. The arithmetic carry that the accumulator generates is used as a pulse to increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 960*.
The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is $2^{32} / 1.32$, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is $2^{32} / 1.30$, or 0xC4EC 4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B 7672. When there is not clock drift, the default addend value of 0xC1F0 7C1F ($2^{32} / 1.32$) must be programmed.

In Figure 960, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$
If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and resynchronize with the master using the new value.

The algorithm is as follows:

1. At time MasterSyncTimeₙ the master sends the slave clock a SYNC message. The slave receives this message when its local clock is SlaveClockTimeₙ and computes MasterClockTimeₙ as follows:
   \[ \text{MasterClockTime}_n = \text{MasterSyncTime}_n + \text{MasterToSlaveDelay}_n \]

2. The master clock counts for current Sync cycle, MasterClockCountₙ is
   \[ \text{MasterClockCount}_n = \text{MasterClockTime}_n - \text{MasterClockTime}_{n-1} \]
   (assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, SlaveClockCountₙ is
   \[ \text{SlaveClockCount}_n = \text{SlaveClockTime}_n - \text{SlaveClockTime}_{n-1} \]

4. The difference between master and slave clock counts for current Sync cycle, ClockDiffCountₙ is
   \[ \text{ClockDiffCount}_n = \text{MasterClockTime}_n - \text{SlaveClockTime}_n \]

5. The frequency-scaling factor for slave clock, FreqScaleFactorₙ is
   \[ \text{FreqScaleFactor}_n = (\text{MasterClockCount}_n + \text{ClockDiffCount}_n) / \text{SlaveClockCount}_n \]

6. The frequency compensation value for Addend register, FreqCompensationValueₙ is
   \[ \text{FreqCompensationValue}_n = \text{FreqScaleFactor}_n \times \text{FreqCompensationValue}_{n-1} \]

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to Section 64.9.9: Programming guidelines for IEEE 1588 timestamping for detailed programming steps.

**IEEE 1588 auxiliary snapshots**

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where \( x = 1 \) to 4) sideband signal.

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: \text{Auxiliary timestamp seconds register (ETH_MACATSSR)} and \text{Auxiliary timestamp nanoseconds register (ETH_MACATSNR)}.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the \text{Timestamp status register (ETH_MACTSSR)} to know the timestamp of which input is available for reading at the top of this FIFO.
When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the Timestamp status register (ETH_MACTSSR). This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in Auxiliary control register (ETH_MACACR). When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of Timestamp status register (ETH_MACTSSR).

**Flexible pulse-per-second output**

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out outputs:

- **Fixed pulse-per-second output**
  
  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).

- **Flexible pulse-per-second output**
  
  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:
  
  The start and stop times are programmed through **PPS target time seconds register (ETH_MACPPSTTSR)** and **PPS target time nanoseconds register (ETH_MACPPSTTNR)**.
  
  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through **PPS width register (ETH_MACPPSWR)** and **PPS interval register (ETH_MACPPSIR)**, respectively.

**Note:** By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.
- **PPS target time seconds register (ETH_MACPPSTTSR)** and **PPS target time nanoseconds register (ETH_MACPPSTTNR)** are used only for generating target time reached interrupt; they are not used for PPS output generation.
- TRGTMODSEL0/1/2/3 must be programmed to 0.
- the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).
Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the Subsecond increment register (ETH_MACSSIR). The pulse width can be programmed from 1 to 232−1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 232−1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

**Note:** The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control Register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR) (as shown in Figure 960).

Refer to Section 64.9.12: Programming guidelines for flexible pulse-per-second (PPS) output for further details on how configuring flexible pulse output.

**PPS start and stop times**

The initial start time can be programmed in the Target Time registers.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of PPS target time nanoseconds register (ETH_MACPPSTTNR) indicates that the synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

**PPS width and interval**

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.
PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through **PTP Offload control register (ETH_MACPOCR)**. 80-bit PTP node identity is configured through the following three registers: **PTP Source Port Identity 0 Register (ETH_MACSPI0R)**, **PTP Source port identity 1 register (ETH_MACSPI1R)** and **PTP Source port identity 2 register (ETH_MACSPI2R)**.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. 

**Table 708** indicates the PTP message generation criteria.

<table>
<thead>
<tr>
<th>SNAPTPYSE</th>
<th>TSMSTRENA</th>
<th>TSEVNTENA</th>
<th>Mode</th>
<th>Criteria for generation of PTP messages</th>
<th>PTP message type generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Ordinary or Boundary Slave</td>
<td>SYNC message reception</td>
<td>Delay_Req</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Ordinary or Boundary Master</td>
<td>Periodic or on trigger from application</td>
<td>SYNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Delay_Resp message reception</td>
<td>Delay_Resp</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
<td>Transparent Slave</td>
<td>Periodic or on trigger from application</td>
<td>Pdelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pdelay_Resp message reception</td>
<td>Pdelay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SYNC message reception</td>
<td>Delay_Req</td>
</tr>
</tbody>
</table>
Table 708. PTP message generation criteria (continued)

<table>
<thead>
<tr>
<th>SNAPTYPSEL</th>
<th>TSMSTREN</th>
<th>TSEVNTENA</th>
<th>Mode</th>
<th>Criteria for generation of PTP messages</th>
<th>PTP message type generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>1</td>
<td>1</td>
<td>Transparent Master</td>
<td>Periodic or on trigger from application</td>
<td>Pdelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Pdelay_Req message reception</td>
<td>Pdelay_Resp</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Periodic or on trigger from application</td>
<td>SYNC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Delay_Req message reception</td>
<td>Delay_Resp</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>Peer-to-Peer</td>
<td>Periodic or on trigger from application</td>
<td>Pdelay_Req</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Transparent</td>
<td>Pdelay_Req message reception</td>
<td>Pdelay_Resp</td>
</tr>
</tbody>
</table>

All other programming combinations are invalid for PTP Offload feature.

Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP Offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received SYNC only when the bit is reset.

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in Table 708.

The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes
them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver.

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

**PTP packet generation**

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Octets</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transportSpecific</td>
<td>messageType</td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td>versionPTP</td>
<td>1</td>
</tr>
<tr>
<td>messageLength</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>domainNumber</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>flagField</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>correctionField</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>sourcePortIdentity</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>sequenceId</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>sequenceId</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>controlField</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>logMessageInterval</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
PTP message header fields

- **messageType**
  The following encoded values are used for PTP message types:
  - SYNC: 0000
  - Delay_Req: 0001
  - Pdelay_Req: 0010
  - Pdelay_Resp: 0011
  - Delay_Resp: 1001

- **transportSpecific**
  The following transport protocol encoding is used:
  - Default PTP over Ethernet: 0000
  - 802.1AS mode: 0001

- **versionPTP**
  It is always set to 2 because PTP version 2 is supported.

- **domainNumber**
  This field contains the value from the PTP Offload control register (ETH_MACPOCR).

- **flagField**
  The following values are used:
  - alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  - twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  - unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address

- **correctionField**
  For more information, see Table 710.

- **sourcePortIdentity**
  This field takes the value programmed in the PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R).

- **sequenceId**
  Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.

- **controlField**
  The following encoded values are used for controlField:
  - SYNC: 0000 0000
  - Delay_Req: 0000 0001
  - Pdelay_Req: 0000 0010
  - Pdelay_Resp: 0000 0101
  - Delay_Resp: 0000 0011

- **logMessageInterval**
  - SYNC:
    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.
Delay_Resp:
This field contains the sum of DRSYNCR and logSyncInterval value taken from the Log message interval register (ETH_MACLMIR) for a multicast PTP message and 0111 1111 for unicast PTP message.

Delay_Req, Pdelay_Req and Pdelay_Resp: 0111 1111
where logSyncInterval = \log_2 (\text{Mean Value of Interval in seconds})

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- $2^{(30+N)}$ ns, when N is negative (–1 to –15)
- $2^N$ seconds, when N is positive (0 to 15)

For example:
- When logSyncInterval is programmed to 1, the interval is $2^1$; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is $2^{-1} = 0.536$ seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because $2^{-30} = 1$ ns.
- When logSyncInterval is programmed to –5, the interval is $2^{-5} = 33.55$ ms; therefore, the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation $2^{(30+N)}$, because of the non-binary nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:

- **messageLength**
  There is no suffix supported, so this field contains the length of the PTP message that includes 34-byte PTP common header and the body specific to the message type. For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.

- **originTimestamp**
  This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req PTP messages.

- **receiveTimestamp**
  For Delay_Resp PTP message, this is the ingress timestamp of the corresponding received Delay_Req PTP message.

- **requestingPortIdentity**
  For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.

- **requestReceiptTimestamp**
  For the Pdelay_Resp PTP message, this field is set to 0.
One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 710 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the Timestamp control Register (ETH_MACTSCR) and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 710. MAC Transmit PTP mode and one-step timestamping operation

<table>
<thead>
<tr>
<th>Programming</th>
<th>Mode</th>
<th>Per packet control(1)</th>
<th>Messages processed on transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TTSE (2)</td>
<td>OSTC (3)</td>
</tr>
<tr>
<td>SNAPTYPSEL</td>
<td>TSMSTR</td>
<td>TSEMST</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENA</td>
<td>ENA</td>
<td></td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>N/A</td>
</tr>
<tr>
<td>00</td>
<td>X</td>
<td>0</td>
<td>End-to-end transparent</td>
</tr>
<tr>
<td>00</td>
<td>0</td>
<td>1</td>
<td>Ordinary or Boundary Slave</td>
</tr>
<tr>
<td>00</td>
<td>1</td>
<td>1</td>
<td>Ordinary or Boundary Master</td>
</tr>
<tr>
<td>01</td>
<td>X</td>
<td>0</td>
<td>End-to-end Transparent with support for peer delay mechanism</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Ingress TS</td>
</tr>
</tbody>
</table>
### Table 710. MAC Transmit PTP mode and one-step timestamping operation (continued)

<table>
<thead>
<tr>
<th>Programming</th>
<th>Mode</th>
<th>Per packet control(1)</th>
<th>Messages processed on transmission</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNAP SEL</td>
<td>TSMSTR ENA</td>
<td>TSEVNT ENA</td>
<td>TTSE (2)</td>
</tr>
<tr>
<td>01 0 1</td>
<td>Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent</td>
<td>0 1</td>
<td>Ingress TS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 1 X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 1 X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 1</td>
</tr>
<tr>
<td>01 1 1</td>
<td>Ordinary or Boundary Master with support for peer delay mechanism</td>
<td>0 1 X</td>
<td>Sync (originTimestamp field) Sync (correction field for subnanosecond correction)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 1 X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 1</td>
</tr>
<tr>
<td>10 X X</td>
<td>End-to-end transparent</td>
<td>0 1</td>
<td>Ingress TS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

Enabling one-step timestamp

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

64.5.5 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the Receive path.

Transmit checksum offload engine

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).
Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

\[
\text{Packet size} < \text{TxQSize} - (\text{PBL} + 7) \times 4
\]

where
- TxQSize corresponds to the TQS bitfield of Tx queue operating mode register (ETH_MTLTXQOMR)
- PBL corresponds to the TxPBL bitfield of Channel transmit control register (ETH_DMACTXCR)

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

**IP header checksum engine**

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

Note: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in Table 732: TDES3 normal descriptor (write-back format)).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as
indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in Table 732: TDES3 normal descriptor (write-back format)). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC Transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 711 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Section: Transmit checksum offload engine because it might result in incorrect checksum insertion or unexpected behavior.
### Table 711. Transmit checksum offload engine functions for different packet types

<table>
<thead>
<tr>
<th>Packet type</th>
<th>Hardware IP header checksum insertion</th>
<th>Hardware TCP/UDP checksum insertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-IPv4 or IPv6 packet</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in <a href="#">Section : Transmit checksum offload engine</a></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in <a href="#">Section : Transmit checksum offload engine</a></td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 with TCP, UDP, or ICMP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 packet has IP options (IP header is longer than 20 bytes)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Packet is an IPv4 fragment</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet with the following next header fields in main or extension headers:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 main header)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 extension header)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Destinations options</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Routing (with segment left 0)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Routing (with segment left &gt; 0)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– TCP, UDP, or ICMP</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Authentication</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Any other next header field in main or extension headers</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has TCP header with Options fields</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 Tunnels:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– IPv4 packet in an IPv4 tunnel</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv4 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv6 Tunnels:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– IPv4 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled)</td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 frames with security features (such as encapsulated security payload)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv6 frames with security features (such as encapsulated security payload)</td>
<td>Not applicable</td>
<td>No</td>
</tr>
</tbody>
</table>
**Receive checksum offload engine**

The Receive Checksum Offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The Receive Checksum Offload Engine (Rx COE) can be enabled by setting the IPC bit of **Operating mode configuration register (ETH_MACCCR)**. When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the **VLAN tag register (ETH_MACVTR)** is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the **Rx queue operating mode register (ETH_MTLRXQOMR)** is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

**Table 712: Receive checksum offload engine functions for different packet types** describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

**Note:** The MAC does not append any payload checksum bytes to the received Ethernet packets.
### Table 712. Receive checksum offload engine functions for different packet types

<table>
<thead>
<tr>
<th>Packet type</th>
<th>Hardware IP header checksum checking</th>
<th>Hardware TCP/UDP/ICMP checksum checking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-IPv4 or IPv6</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)</td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 with TCP, UDP, or ICMP</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 packet's protocol field contains a protocol other than TCP, UDP, or ICMP</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv4 packet has IP options (IP header is longer than 20 bytes)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Packet is an IPv4 fragment</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv6 packet with the following next header fields in main or extension headers:</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 main header)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Hop-by-hop options (in IPv6 extension header)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Destinations options</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Routing (with segment left 0)</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– Routing (with segment left &gt; 0)</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– TCP, UDP, or ICMP</td>
<td>– Not applicable</td>
<td>– Yes</td>
</tr>
<tr>
<td>– Any other next header field in main or extension headers</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has TCP header with Options fields</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 Tunnels:</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv4 packet in an IPv4 tunnel</td>
<td>– Yes (IPv4 tunnel header)</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv4 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv6 Tunnels:</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv4 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>– IPv6 packet in an IPv6 tunnel</td>
<td>– Not applicable</td>
<td>– No</td>
</tr>
<tr>
<td>IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled)</td>
<td>Not applicable</td>
<td>Yes</td>
</tr>
<tr>
<td>IPv4 frames with security features (such as encapsulated security payload)</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IPv6 frames with security features (such as encapsulated security payload)</td>
<td>Not applicable</td>
<td>No</td>
</tr>
</tbody>
</table>
64.5.6 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in Figure 961.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACCR register (see Channel transmit control register (ETH_DMACTXCR)). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to Section 64.9.13: Programming guidelines for TSO.

**Figure 961. TCP segmentation offload overview**
DMA operation with TSO feature

*Figure 962* shows the TSO flow.

**Figure 962. TCP segmentation offload flow**

For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA Tx channel.
3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   - If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.
- If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.
- If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.

5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.

6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.

7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.

8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.

9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.

10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.
TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 713* describes how the TCP and IP headers are updated.

<table>
<thead>
<tr>
<th>Packet sequence</th>
<th>TCP header</th>
<th>IP header</th>
</tr>
</thead>
</table>
| **First packet** | 1. The sequence number is not updated. The value provided in the header is used.  
2. If set, the FIN and PSH flags are cleared.  
3. The TCP checksum is calculated again. | IPv4 Header  
– Total Length = MSS + TCP  
Header Length + IP Header Length  
– Identification field is not modified. It is sent as per the header provided by the software.  
– IPv4 Header Checksum is recalculated  
IPv6 Header  
– Payload Length = MSS + TCP  
Header Length + IP Extension Header Length |
| **Subsequent packets** | 1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment.  
2. If set, the FIN and PSH flags are cleared.  
3. The TCP checksum is calculated again. | IPv4 Header  
– Total Length = MSS + TCP  
Header Length + IP Header Length  
– Identification field = Previous Identification Field + 1  
– IPv4 Header Checksum is recalculated  
IPv6 Header  
– Payload Length = MSS + TCP  
Header Length + IP Extension Header Length |
| **Last packet** | 1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment.  
2. If FIN and PSH flags were set in original header, these flags are set.  
3. The TCP checksum is calculated again. | IPv4 Header  
– Total Length = Remaining  
Payload + TCP Header Length + IP Header Length  
– Identification Field = Previous Identification Field + 1  
– IPv4 header Checksum is recalculated  
IPv6 Header  
– Payload Length = Remaining  
Payload Length + TCP Header Length + IP Extension Header Length |
Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in Table 713: TSO: TCP and IP header fields. Figure 963: Header and payload fields of segmented packets shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 963. Header and payload fields of segmented packets

Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to Section 64.10.3: Transmit descriptor).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.
If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see **Channel interrupt enable register (ETH_DMAMEN)**).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see **Channel status register (ETH_DMACSR)**).

**Building the Descriptor and the packet for the TSO feature**

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see **Section 64.10.3: Transmit descriptor**). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer 1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMAMCR (see **Channel control register (ETH_DMAMCR)**) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACTXCR register (see **Channel transmit control register (ETH_DMACTXCR)**) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQOMR register (see **Tx queue operating mode register (ETH_MTLTXQOMR)**). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.
If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

64.5.7 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets. ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:
1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   - DA field of the Ethernet packet header
   - Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   - SA field of the Ethernet packet header
   - Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply.

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the Extended operating mode configuration register (ETH_MACECR) is set, then the MAC does not check for valid CRC of a ARP
request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

64.5.8 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using loopback mode

Below some guidelines for using the loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The Transmit and Receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the Receive path. The FIFO is free-running to write on the write clock (eth_tx_clk) and read on every read clock (eth_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of four (in 10/100 Mbps mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between MII Transmit and Receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the Receive Protocol Engine module generates received packet status and sends it to the Receive Packet Controller module. The control, missed packet, and filter fail status are added to the Receive status in the Receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling loopback mode

To enable this feature, program the LM bit of the Operating mode configuration register (ETH_MACCR). Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal Receive MII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on mii_txd signal.

Preemption is not supported in loopback mode.
64.5.9 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and back-pressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

**Flow control in Full-duplex mode**

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. Table 714 describes the fields of a Pause packet.

### Table 714. Pause packet fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA</td>
<td>Contains the special multicast address</td>
</tr>
<tr>
<td>SA</td>
<td>Contains the MAC address 0</td>
</tr>
<tr>
<td>Type</td>
<td>Contains 8808</td>
</tr>
<tr>
<td>MAC Control opcode</td>
<td>Contains 0001 for IEEE 802.3x Pause Control packets</td>
</tr>
<tr>
<td>PT</td>
<td>Contains Pause time specified in the PT field of the Tx Queue flow control register (ETH_MACQTXFCR)</td>
</tr>
</tbody>
</table>

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

**Flow control in Half-duplex mode**

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

*Table 715* describes the flow control in the Tx path based on the setting of the following bits:

- TFE bit of *Tx Queue flow control register (ETH_MACQTXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.
Transmit flow control

The transmit flow control is enabled when TFE bit is set in Transmission queue flow control register (ETH_MACQTXFCR).

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding Transmit queue flow control register (ETH_MACQTXFCR).

Receive flow control

In the Receive path, the flow control is functional only in Full-duplex mode. If any Pause packet is received in Half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 716 describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of Receive flow control register (ETH_MACRXFCR)
- DM bit of Operating mode configuration register (ETH_MACCR)

<table>
<thead>
<tr>
<th>TFE</th>
<th>DM</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x</td>
<td>The MAC transmitter does not perform the flow control or backpressure operation.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>The MAC transmitter performs backpressure when Bit 0 of Transmit queue flow control register (ETH_MACQTXFCR) is set.</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>The MAC transmitter sends the Pause packet when Bit 0 of Transmit queue flow control register (ETH_MACQTXFCR) is set.</td>
</tr>
</tbody>
</table>

Table 715. Tx MAC flow control

Table 716. Rx MAC flow control

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   - Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   - Unicast destination address: the DA matches the content of the MAC Address 0 registers (MAC Address 0 high register (ETH_MACA0HR) and MAC Address X...
low register (ETH_MACAxLR)) and the UP bit of Rx flow control register (ETH_MACRXFCR) is set.

If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.

2. The MAC decodes the following fields of the received packet:
   - Type field: this field is checked for 0x8808.
   - Opcode field: this field is checked for 0x0001 (Pause packet).
   - Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.

3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

**Enabling receive flow control**

Set the RFE bit in the Rx flow control register (ETH_MACRXFCR) to enable the Pause flow control.
64.5.10 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in Section 64.11.4: Ethernet MAC and MMC registers.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The Receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the Packet filtering control register (ETH_MACPFR). The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- Six registers used for collision, error, and good packet counters:
  - Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)
  - Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)
  - Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)
  - Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)
  - Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)
  - Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)

- Four registers to record LPI mode transition:
  - Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)
  - Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)
  - Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)
  - Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)
Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision

- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbps only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)

- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022

- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the Table 717.

### Table 717. Size of the maximum receive packet

<table>
<thead>
<tr>
<th>JE</th>
<th>S2KP</th>
<th>GPSLCE</th>
<th>EDVLP</th>
<th>Untagged frame maximum size in bytes</th>
<th>Single VLAN frame maximum size in bytes</th>
<th>Double VLAN Frame maximum size in bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>9018</td>
<td>9022</td>
<td>9026</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>2000</td>
<td>2000</td>
<td>2000</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>GPSL</td>
<td>GPSL+4</td>
<td>GPSL+8</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1518</td>
<td>1522</td>
<td>1526</td>
</tr>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>9018</td>
<td>9022</td>
<td>9022</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>GPSL</td>
<td>GPSL+4</td>
<td>GPSL+4</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1518</td>
<td>1522</td>
<td>1522</td>
</tr>
</tbody>
</table>
64.5.11 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The interrupt status register (ETH_MACISR) describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the interrupt enable register (ETH_MACIER).

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

64.5.12 MAC and MMC register descriptions

Refer to Section 64.11.4: Ethernet MAC and MMC registers.
64.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII one. All other interfaces are derived from it as shown in Figure 964.

Figure 964. Supported PHY interfaces

This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

- Station management agent (SMA)
- Media independent interface (MII)
- Reduced media independent interface (RMII)

64.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in Figure 965.

Figure 965. SMA Interface block
SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the MDIO address register (ETH_MACMDIOAR) register. The MDC clock is selected as follows:

<table>
<thead>
<tr>
<th>Selection</th>
<th>MDC clock</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>eth_hclk/42</td>
</tr>
<tr>
<td>0001</td>
<td>eth_hclk/62</td>
</tr>
<tr>
<td>0010</td>
<td>eth_hclk/16</td>
</tr>
<tr>
<td>0011</td>
<td>eth_hclk/26</td>
</tr>
<tr>
<td>0100</td>
<td>eth_hclk/102</td>
</tr>
<tr>
<td>0101</td>
<td>eth_hclk/124</td>
</tr>
<tr>
<td>0110, 0111</td>
<td>-</td>
</tr>
</tbody>
</table>

The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

*Figure 966* shows the structure of a Clause 45 MDIO packet, while *Table 719* provides a detailed description of the packet fields.

**Figure 966. MDIO packet structure (Clause 45)**
The frame structure for Clause 22 frames is also supported. The C45E bit in MDIO address register (ETH_MACMDIOAR) can be programmed to enable Clause 22 or Clause 45 mode of operation. Figure 967 shows the structure of a Clause 22 MDIO packet, while Table 720 provides a detailed description of the packet fields.

**Table 719. MDIO Clause 45 frame structure**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>The ETH_MDIO line is three-state; there is no clock on ETH_MDC.</td>
</tr>
<tr>
<td>PREAMBLE</td>
<td>32 continuous bits of value 1</td>
</tr>
<tr>
<td>START</td>
<td>Start of packet is 0b00</td>
</tr>
<tr>
<td>OPCODE</td>
<td>– 0b00: Address</td>
</tr>
<tr>
<td></td>
<td>– 0b01: Write</td>
</tr>
<tr>
<td></td>
<td>– 0b10: Read+ Address</td>
</tr>
<tr>
<td></td>
<td>– 0b11: Read</td>
</tr>
<tr>
<td>PHY ADDR</td>
<td>5-bit address select for one of 32 PHYs</td>
</tr>
<tr>
<td>DEV TYPE</td>
<td>5-bit device type</td>
</tr>
<tr>
<td>TA</td>
<td>Tumaround</td>
</tr>
<tr>
<td></td>
<td>– 0bZ0: Read and post-read increment address</td>
</tr>
<tr>
<td></td>
<td>– 0b10: Write and address MDIO accesses where Z is the tri-state level</td>
</tr>
<tr>
<td>DATA</td>
<td>16-bit value: for an address cycle (OPCODE = 0b00), this frame contains</td>
</tr>
<tr>
<td></td>
<td>the address of the register to be accessed on the next cycle. For the data</td>
</tr>
<tr>
<td></td>
<td>cycle of a write frame, this field contains the data to be written to the</td>
</tr>
<tr>
<td></td>
<td>register. For read or post-read increment address frames, this field</td>
</tr>
<tr>
<td></td>
<td>contains the contents of the register read from the PHY.</td>
</tr>
<tr>
<td></td>
<td>– In address and data write cycles, the peripheral drives the ETH_MDIO</td>
</tr>
<tr>
<td></td>
<td>line during the transfer of these 16 bits.</td>
</tr>
<tr>
<td></td>
<td>– In read and post-read increment address cycles, the PHY drives the</td>
</tr>
<tr>
<td></td>
<td>ETH_MDIO line during the transfer of these 16 bits.</td>
</tr>
</tbody>
</table>

The frame structure for Clause 22 frames is also supported. The C45E bit in MDIO address register (ETH_MACMDIOAR) can be programmed to enable Clause 22 or Clause 45 mode of operation. Figure 967 shows the structure of a Clause 22 MDIO packet, while Table 720 provides a detailed description of the packet fields.

**Figure 967. MDIO packet structure (Clause 22)**

```markdown
<table>
<thead>
<tr>
<th>IDLE</th>
<th>PREAMBLE</th>
<th>START</th>
<th>OPCODE</th>
<th>PHY ADDR</th>
<th>REG ADDR</th>
<th>TA</th>
<th>DATA</th>
</tr>
</thead>
</table>
```
In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

**MII management write operations**

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

*Figure 968* illustrates the flow for a write operation from the SMA module to the PHY registers.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IDLE</td>
<td>The ETH_MDIO line is three-state; there is no clock on ETH_MDC.</td>
</tr>
<tr>
<td>PREAMBLE</td>
<td>32 continuous bits of value 1</td>
</tr>
<tr>
<td>START</td>
<td>Start of packet is 0b01</td>
</tr>
<tr>
<td>OPCODE</td>
<td>0b10 for Read and 0b01 for Write</td>
</tr>
<tr>
<td>PHY ADDR</td>
<td>5-bit address select for one of 32 PHYs</td>
</tr>
<tr>
<td>REG ADDR</td>
<td>Register address in the selected PHY</td>
</tr>
<tr>
<td>TA</td>
<td>Turnaround</td>
</tr>
<tr>
<td></td>
<td>- 0bZ0: Read and post-read increment address</td>
</tr>
<tr>
<td></td>
<td>- 0b10: Write and address MDIO accesses</td>
</tr>
<tr>
<td></td>
<td>where Z is the tri-state level</td>
</tr>
<tr>
<td>DATA</td>
<td>Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.</td>
</tr>
</tbody>
</table>
When bits[3:2] are set to 01 and bit 0 to 1 in the MDIO address register (ETH_MACMDIOAR), the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (MDIO data register (ETH_MACMDIODR)) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the MII management interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, Section 22.2.4.5).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations performed to the MDIO address register (ETH_MACMDIOAR) or the MDIO data register.
(ETH_MACMDIODR) during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

**Figure 969. Write data packet**

<table>
<thead>
<tr>
<th>IDLE</th>
<th>PREAMBLE</th>
<th>START</th>
<th>OPCODE</th>
<th>PHY ADDR</th>
<th>REG ADDR</th>
<th>TA</th>
<th>DATA</th>
<th>IDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1111..11</td>
<td>01</td>
<td>01</td>
<td>AAAA</td>
<td>RRRRR</td>
<td>10</td>
<td>DDD...DDD</td>
<td>Z</td>
</tr>
</tbody>
</table>

**MII management read operation**

When bits[3:2] are set to 11 and bit 0 to 1 in the MDIO address register (ETH_MACMDIOAR), the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the MII management interface using the management packet format specified in the MII specifications (as per IEEE 802.3-2002 specifications, Section 22.2.4.5).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the MDIO address register (ETH_MACMDIOAR) or MDIO data register (ETH_MACMDIODR) during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the MDIO data register (ETH_MACMDIODR) with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

**Figure 970. Read data packet**

<table>
<thead>
<tr>
<th>IDLE</th>
<th>PREAMBLE</th>
<th>START</th>
<th>OPCODE</th>
<th>PHY ADDR</th>
<th>REG ADDR</th>
<th>TA</th>
<th>DATA</th>
<th>IDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>1111..11</td>
<td>01</td>
<td>10</td>
<td>AAAA</td>
<td>RRRRR</td>
<td>Z0</td>
<td>DDD...DDD</td>
<td>Z</td>
</tr>
</tbody>
</table>
Preamble suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in MDIO address register (ETH_MACMDIOAR).

Trailing clocks and back-to-back transactions

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in MDIO address register (ETH_MACMDIOAR) allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in MDIO address register (ETH_MACMDIOAR) when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

Interrupt for MDIO transaction completion

The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the MDIO address register (ETH_MACMDIOAR) to know the completion of MDIO commands.
64.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in Figure 971: Media independent interface (MII) signals.

- **TX_CLK**: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- **TXD[3:0]**: transmit data.
  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- **TX_EN**: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- **TX_ER (optional)**: required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- **RX_CLK**: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- **RXD[3:0]**: receive data
  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is
the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific
RXD[3:0] value is used to transfer specific information from the PHY.

- **RX_DV**: receive data valid
  This signal indicates that the PHY is presenting recovered and decoded nibbles on the
  MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered
  nibble of the frame and must remain asserted through the final recovered nibble. It
  must be deasserted prior to the first clock cycle that follows the final nibble. In order
to receive the frame correctly, the RX_DV signal must encompass the frame, starting no
later than the SFD field.

- **RX_ER**: receive error
  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the
  MAC sublayer that an error was detected somewhere in the frame. This error condition
  must be qualified by RX_DV assertion.

- **CRS**: carrier sense.
  This signal is asserted by the PHY when either the transmit or receive medium is non
  idle. It is deasserted by the PHY when both transmit and receive media are idle. The
  PHY must ensure that the CS signal remains asserted throughout the duration of a
  collision condition. This signal is not required to transition synchronously with respect
to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don’t care for the
  MAC sublayer.

- **COL**: collision detection signal
  This signal must be asserted by the PHY upon detection of a collision on the medium
  and must remain asserted while the collision condition persists. This signal is not
  required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex
  mode, the state of this signal is don’t care for the MAC sublayer.

### Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count
between Ethernet PHYs and STM32 MCU. According to the IEEE 802.3u, an MII contains
16 pins for data and control. RMII specification reduces the pin count to 7.

Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This
helps in translating the MII of the MAC into the RMII. The RMII block has the following
characteristics:

- **Supports 10 Mbps and 100 Mbps operating rates. It does not support the 1000 Mbps
  operation.**
- **Provides independent 2-bits wide Transmit and Receive paths by sourcing two clock
  references externally.**
RMII block diagram

*Figure 972: RMII block diagram* shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

**Figure 972. RMII block diagram**

- **RMII_REF_CLK**: continuous 50 MHz reference clock input
- **TXD[1:0]**: transmit data
- **TX_EN**: transmit data enable. When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- **RXD[1:0]**: receive data
- **CRS_DV**: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in Figure 973: Transmission bit order. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

Figure 973. Transmission bit order
Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in Figure 974: Receive bit order. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 974. Receive bit order
64.7 Ethernet low-power modes

64.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is in low-power mode, and has to be woken up only at the reception of specific packets from the Ethernet network. In low-power mode, the clocks on most of the host logic, along with the majority of the peripherals (except the MAC receiver logic), can be disabled. On receiving the specific packets from the network, the MAC generates the trigger to wake up the host system and come back to normal state. Refer to the power control (PWR) section of the reference manual for the list of the system operating modes that support Ethernet low-power modes.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and burst traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of PMT control status register (ETH_MACPCSR) is programmed to 1. Exit from the magic-packet-based power saving mode is enabled by setting the MGPKTEN bit of PMT control status register (ETH_MACPCSR) to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register (ETH_MACA0LR)) or with multicast/broadcast address.
The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.

The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

**Magic packet data format**

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (MAC Address 0 high register (ETH_MACA0HR) and the MAC Address 0 low register (ETH_MACA0LR)) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

```
Destination Address Source Address Length/Type............... FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 ...CRC
```

**Description of remote wake-up packet mode**

This section describes the power saving mode based on remote wake-up packet.

*Note:* The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in PMT control status register (ETH_MACPCSR) is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of PMT control status register (ETH_MACPCSR) to 1.

The MAC implements a filter lookup table (programmed through Remote wake-up packet filter register (ETH_MACRWKPFR) in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register (ETH_MACA0LR)) or with multicast/broadcast address.
• The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
• The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in **PMT control status register (ETH_MACPCSR)** and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in **Interrupt status register (ETH_MACISR)** is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in **Interrupt enable register (ETH_MACIER)** is set) and CSR clock is not gated off in low-power mode.

**Remote wake-up packet filters**

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in **Table 721: Remote wake-up packet filter register**.

**Table 721. Remote wake-up packet filter register**

<table>
<thead>
<tr>
<th>ETH_MACRWPFR value</th>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Filter 0 byte mask</td>
</tr>
<tr>
<td>1</td>
<td>Filter 1 byte mask</td>
</tr>
<tr>
<td>2</td>
<td>Filter 2 byte mask</td>
</tr>
<tr>
<td>3</td>
<td>Filter 3 byte mask</td>
</tr>
<tr>
<td>4</td>
<td>Filter 3 command offset</td>
</tr>
<tr>
<td>5</td>
<td>Filter 1 command offset</td>
</tr>
<tr>
<td>6</td>
<td>Filter 0 command offset</td>
</tr>
<tr>
<td>7</td>
<td>Filter 1 CRC - 16</td>
</tr>
<tr>
<td></td>
<td>Filter 0 CRC - 16</td>
</tr>
<tr>
<td></td>
<td>Filter 2 CRC - 16</td>
</tr>
</tbody>
</table>

The remote wake-up filter fields are described in **Table 722: Description of the remote wake-up filter fields**.
Table 722. Description of the remote wake-up filter fields

<table>
<thead>
<tr>
<th>Register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter i Byte mask</td>
<td>The filter i byte mask register defines the bytes of the packet that are examined by filter i (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet.</td>
</tr>
<tr>
<td></td>
<td>– The MSB (31st bit) must be zero.</td>
</tr>
<tr>
<td></td>
<td>– Bit j[30:0] is the byte mask.</td>
</tr>
<tr>
<td></td>
<td>– If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter i Offset + j of the incoming packet; otherwise Filter i Offset + j is ignored.</td>
</tr>
<tr>
<td>Filter i Command</td>
<td>The 4-bit filter i command controls the filter i operation.</td>
</tr>
<tr>
<td></td>
<td>– Bit 3 specifies the address type, defining the destination address type of the pattern.</td>
</tr>
<tr>
<td></td>
<td>When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet.</td>
</tr>
<tr>
<td></td>
<td>– Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value.</td>
</tr>
<tr>
<td></td>
<td>Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as &quot;Pattern 1 AND NOT Pattern 2&quot;.</td>
</tr>
<tr>
<td></td>
<td>– Bit 1 (And_Previous) implements the Boolean logic(1).</td>
</tr>
<tr>
<td></td>
<td>When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set.</td>
</tr>
<tr>
<td></td>
<td>– Bit 0 is the enable for filter i. If Bit 0 is not set, filter i is disabled.</td>
</tr>
<tr>
<td>Filter i Offset</td>
<td>This filter i offset register defines the offset (within the packet) from which the filter i examines the packets.</td>
</tr>
<tr>
<td></td>
<td>– This 8-bit pattern-offset is the offset for the filter i first byte to be examined.</td>
</tr>
<tr>
<td></td>
<td>– The minimum allowed offset is 12, which refers to the 13th byte of the packet.</td>
</tr>
<tr>
<td></td>
<td>– The offset value 0 refers to the first byte of the packet.</td>
</tr>
<tr>
<td>Filter i CRC-16</td>
<td>This filter i CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block.</td>
</tr>
<tr>
<td></td>
<td>– The 16-bit CRC calculation uses the following polynomial: G(x) = x^16 + x^15 + x^2 + 1</td>
</tr>
<tr>
<td></td>
<td>Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following:</td>
</tr>
<tr>
<td></td>
<td>– 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1, the corresponding byte is taken into the CRC16 calculation.</td>
</tr>
<tr>
<td></td>
<td>– 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation.</td>
</tr>
<tr>
<td></td>
<td>The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.</td>
</tr>
</tbody>
</table>

1. The And_Previous bit setting is applicable within a set of four filters. Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect. If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered. If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.
The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through Remote wake-up packet filter register (ETH_MACRWKPFR). The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in Remote wake-up packet filter register (ETH_MACRWKPFR) for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of PMT control status register (ETH_MACPCSR).

Note: If the Remote wake-up packet filter register (ETH_MACRWKPFR) is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

When Remote wake-up packet filter register (ETH_MACRWKPFR) is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the Remote wake-up packet filter register (ETH_MACRWKPFR) until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the Remote wake-up packet filter register (ETH_MACRWKPFR) should be at least 4 cycles of the PHY receive clock.

PMT interrupt

The PMT interrupt signal is asserted when a valid remote wake-up packet is received. Table 723 lists the remote wake-up scenarios in which PMT interrupt is generated.

<table>
<thead>
<tr>
<th>Filter i Command</th>
<th>Frame Type and CRC Status</th>
<th>Interrupt Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAST</td>
<td>INV</td>
<td>EN</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. In all other combinations, the Remote Wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode. It is ORed with lpi_intr_o signal (see Section : LPI interrupt) and tied to the EXTI peripheral.

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the PMT control status register (ETH_MACPCSR) is read. This is
because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode. When the application clears the PWRDWN bit in Remote wake-up packet filter register (ETH_MACRWKPFR), the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the Channel transmit control register (ETH_DMACTXCR).
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in Debug register (ETH_MACDR) and TXQSTS bit in Tx queue debug register (ETH_MTLTXQDR) of all MTL Tx Queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in Operating mode configuration register (ETH_MACCR).
- Wait till the Receive DMA empties all frames from the Rx FIFO. You can check this by reading PRXQ[13:0] in Rx queue debug register (ETH_MTLRXQDR) of all Rx Queues. If these bits are zero, it indicates that the Rx FIFO is empty.
- Configure the magic packet (MGKPKTEN) and/or remote wake-up (RWKPKTEN) detection in the PMT control status register (ETH_MACPCSR).
- Set bit 31 (ARPEN) in the Operating mode configuration register (ETH_MACCR).
- Enable the MAC Receiver by setting RE bit and then set PWRDWN bit in the PMT control status register (ETH_MACPCSR) to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC Transmitter is in the LPI mode when it is put into the power-down mode, then the MII interface gets clamped to assert the LPI pattern. If the MAC Transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  - De-assert the pmt_intr_o by reading the PMT control status register (ETH_MACPCSR).
  - Perform a write operation (with reset values) to the PMT control status register (ETH_MACPCSR) and the Remote wake-up packet filter register (ETH_MACRWKPFR) so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  - Perform write operations to the Operating mode configuration register (ETH_MACCR), MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register (ETH_MACA0LR) to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.
After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

64.7.2 Energy Efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sub layer along with a family of physical layers to operate in the Low-Power Idle (LPI) mode. The EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbps. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the LPI control and status register (ETH_MACLCSR) to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of LPI timers control register (ETH_MACLTCR). The PHY Link Status PLS bit of the LPI control and status register (ETH_MACLCSR) indicates the link status of the PHY.

Note: The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbps)
4. Updates the status (TLPIEN bit of LPI control and status register (ETH_MACLCSR)) and generates an interrupt.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.
To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:
   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the LPI timers control register (ETH_MACLT) and generates an interrupt.
3. Updates the LPI exit status (TLPIEX bit of the LPI control and status register (ETH_MACCSR)) and generates an interrupt.

Figure 975 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

**Note:** The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 975) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACCSR) and can be done automatically as shown on Figure 976.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

**Figure 975. LPI transitions (Transmit, 100 Mbds)**
Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the LPI control and status register (ETH_MACLCSR).

When LPITXA and LPIEN of LPI control and status register (ETH_MACLCSR) are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in LPI entry timer register (ETH_MACLETR). In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path Functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbps).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the LPI control and status register (ETH_MACLCSR) and immediately generates an interrupt.
Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the LPI control and status register (ETH_MACLCSR) and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 977 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 977. LPI transitions (receive, 100 Mbps)

Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 977.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.
LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the LPI timers control register (ETH_MACLTCR) and LPI entry timer register (ETH_MACLETR):

- **LPI LS TIMER**
  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the LPI timers control register (ETH_MACLTCR). The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.

- **LPI TW TIMER**
  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of LPI timers control register (ETH_MACLTCR). The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count. The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.

- **LPI AUTO ENTRY TIMER**
  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC Transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in LPI control and status register (ETH_MACLCSR) is set.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the LPI control and status register (ETH_MACLCSR).

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state. It is ORed with pmt_intr_o signal (see Section : PMT interrupt) and tied to the EXTI peripheral.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the LPI control and status register (ETH_MACLCSR) is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbps mode.

Programming guidelines for Energy Efficient Ethernet

For detailed guidelines on the programming guidelines, see Section 64.9.11: Programming guidelines for Energy Efficient Ethernet (EEE) on page 3371.
64.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the Interrupt status register (ETH_DMAISR) register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMACSR: Channel Status (see Channel status register (ETH_DMACSR))
- ETH_MTLISR: Interrupt Status (see Interrupt status register (ETH_MTLISR))
- ETH_MACISR: Interrupt Status (see Interrupt status register (ETH_MACISR))

64.8.1 DMA interrupts

Interrupt registers description

The ETH_DMACSR: Channel Status register (see Channel status register (ETH_DMACSR)) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMACIER: Channel Interrupt Enable register (see Channel interrupt enable register (ETH_DMACIER)) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts. They are indicated by Bits[15:14] of ETH_DMACSR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the Interrupt status register (ETH_DMAISR) for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of Interrupt status register (ETH_DMAISR) are cleared.

Periodic scheduling of Transmit and Receive Interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in Table 727: TDES2 normal descriptor (read format)) once for every “required” number of packets to be transmitted.

2. Similarly, set the IOC (RDES3[30] in Table 740: RDES3 normal descriptor (read format)) bit only at some specific intervals of Receive descriptors. This way, whenever a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.
In addition to above, an interrupt timer (ETH_DMACRXIWTR: Channel Rx Interrupt
Watchdog Timer) is given for flexible control and periodic scheduling of Receive Interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive Interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in Table 740: RDES3 normal descriptor (read format)) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMACIER register (see Channel interrupt enable register (ETH_DMACIER)). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor’s IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Channel transfer complete interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (Channel status register (ETH_DMACSR)). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes a descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30]).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (Channel interrupt enable register (ETH_DMACIER)).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (DMA mode register (ETH_DMAMR)). Table 724 explains the behavior of the Transfer Complete interrupt.

<table>
<thead>
<tr>
<th>Interrupt Mode</th>
<th>Behavior of TI/RI and interrupt signal</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTM=0</td>
<td>The TI/RI status signals are set whenever the Transfer complete event is detected. These bits are cleared whenever the software driver writes 1 to these bits. The interrupt signal is asserted whenever the corresponding interrupts are also enabled in ETH_DMACIER register.</td>
</tr>
<tr>
<td>INTM=1</td>
<td>The TI/RI is set as explained above. However, the interrupt is not asserted for any RI/TI event.</td>
</tr>
<tr>
<td>INTM=2</td>
<td>The RI/TI status bits are set whenever the Transfer Complete event is detected and are reset whenever software driver clears them by writing 1. However, if another Transfer complete event is detected before it is cleared (serviced) by the software, then these status bits are automatically set again. However, the interrupt is not generated based on TI/RI.</td>
</tr>
</tbody>
</table>
64.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status register (ETH_MTLISR)* report the queue number responsible for the event. *ETH_MTLQICSR: Queue Interrupt Control Status* must be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events:
- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

64.8.3 MAC Interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:
- Receive Status Interrupt
- Transmit Status Interrupt
- Timestamp Interrupt Status
- MMC Interrupt Status
  - MMC Receive Checksum Offload Interrupt Status
  - MMC Transmit Interrupt Status
  - MMC Receive Interrupt Status
- LPI Interrupt Status
- PMT Interrupt Status
- PHY Interrupt

*Note:* Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.
64.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see Section 64.9.1)
- MTL initialization (see Section 64.9.2)
- MAC initialization (see Section 64.9.3)
- Performing normal receive and transmit operation (see Section 64.9.4)
- Stopping and starting transmission (see Section 64.9.5)
- Programming guidelines for MII link state transitions (see Section 64.9.6)
- Programming guidelines for Energy Efficient Ethernet (see Section 64.9.7)
- Programming guidelines for IEEE 1588 timestamping (see Section 64.9.8)
- Programming guidelines for flexible pulse-per-second (PPS) output (see Section 64.9.9)
- Programming guidelines for MII link state transitions (see Section 64.9.10)
- Programming guidelines for VLAN filtering on the receiver (see Section 64.9.11)
- Programming guidelines for TSO (see Section 64.9.12)
- Programming guidelines for VLAN filtering on the receiver (see Section 64.9.13)
- Programming guidelines for TSO (see Section 64.9.14)

64.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of DMA mode register (ETH_DMAMR)).
2. Wait for the completion of the reset process (poll bit 0 of the DMA mode register (ETH_DMAMR), which is cleared when the reset operation is completed).
3. Program the following fields to initialize the System bus mode register (ETH_DMASBMR):
   a) AAL
   b) Fixed burst or undefined burst
   c) Burst mode values in case of AHB bus interface.
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to Section 64.10: Descriptors.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMACTXRLR and ETH_DMCRXLRLR registers (see Channel Tx descriptor ring length register (ETH_DMACTXRLR) and Channel Rx descriptor ring length register (ETH_DMCRXLRLR)). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (Channel Tx descriptor list address register (ETH_DMACTXDLAR), Channel Rx descriptor list address register (ETH_DMCRXLAR)). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see Channel Tx descriptor tail pointer register (ETH_DMACTXTPR) and Channel Rx descriptor tail pointer register (ETH_DMCRXTTPR)).
7. Program ETH_DMACC, ETH_DMACTXCR and ETH_DMCRXCR registers (see Channel control register (ETH_DMACC), Channel transmit control register (ETH_DMACTXCR) and Channel receive control register (ETH_DMCRXCR)) to
configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.

8. Enable the interrupts by programming the ETH_DMACIER register (see Channel interrupt enable register (ETH_DMACIER)).

9. Start the Receive and Transmit DMAs by setting SR (bit 0) of Channel receive control register (ETH_DMACRXCR) and ST (bit 0) of the ETH_DMACTXCR (see Channel transmit control register (ETH_DMACTXCR)).

64.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the following fields to initialize the operating mode in Tx queue operating mode register (ETH_MTLTXQOMR).
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2'b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).

2. Program the following fields to initialize the operating mode in the ETH_MTLRXQOMR register (see Rx queue operating mode register (ETH_MTLRXQOMR)):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).

64.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: MAC Address x low register (ETH_MACAxLR), MAC Address 0 high register (ETH_MACA0HR) and MAC Address x high register (ETH_MACAxHR).

2. Program the following fields to set the appropriate filters for the incoming frames in the Packet filtering control register (ETH_MACPFR):
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.

3. Program the following fields for proper flow control in the Tx Queue flow control register (ETH_MACQTXFCR):
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.
   c) Flow Control Busy.
4. Program the Interrupt enable register (ETH_MACIER) as required, if it is applicable for your configuration.

5. Program the appropriate fields in the Operating mode configuration register (ETH_MACCR) register. For example, Inter-packet gap while transmission and jabber disable.

6. Set bit 0 and 1 in Operating mode configuration register (ETH_MACCR) register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:
- In the Operating mode configuration register (ETH_MACCR)
  a) Set JE bit to 1.
  b) Set JD and WD bits to 0 to avoid giant packet error reporting.
  c) Set GPSLCE bit to 1
  d) Set GPSL bitfield of the Extended operating mode configuration register (ETH_MACECR) to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:
- In the Operating mode configuration register (ETH_MACCR)
  a) Set JD and WD bits to 1 to avoid giant packet error reporting.
  b) Set GPSLCE bit to 1.
  c) Set GPSL bitfield of the Extended operating mode configuration register (ETH_MACECR) to 16383.

64.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:
1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).

2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.

3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMACTXDTPR (see Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR)) and ETH_DMACRXDTPR (see Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR)).

4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see Channel current application transmit descriptor register (ETH_DMACCATXDR) and Channel current application receive descriptor register (ETH_DMACCARXDR)).

5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMACCATXDR and ETH_DMACCARXDR registers (see Channel current application transmit descriptor register (ETH_DMACCATXDR) and Channel current application receive descriptor register (ETH_DMACCARXDR)).
64.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMACTXCR register (see Channel transmit control register (ETH_DMACTXCR)).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of Tx queue debug register (ETH_MTLTXQDR) (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the Operating mode configuration register (ETH_MACCR) Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of Tx queue debug register (ETH_MTLTXQDR), PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in Tx queue debug register (ETH_MTLTXQDR) and RXQSTS[1:0] is set to 00.
6. To restart the operation, first start the DMAs, and then enable the MAC Transmitter and Receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active. Similarly, do not change the DMA-related configuration when Transmit and Receive DMA are active.

64.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an RxFIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

64.9.7 Programming guidelines for switching the AHB clock frequency

To dynamically change the AHB clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the Tx queue debug
register (ETH_MTLTXQDR) and the status of DMA is given in Debug status register (ETH_DMADSR).

2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in Operating mode configuration register (ETH_MACCR).

3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in Rx queue debug register (ETH_MTLRXQDR).

4. Ensure that the application does not perform any register read or write operation.

5. Change the frequency of the AHB clock.

6. Enable the MAC Transmitter or the MAC Receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

64.9.8 Programming guidelines for MII link state transitions

Transmit and Receive clocks are running when the link is down

Complete the following steps when the link is down while the Transmit and Receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of Channel control register (ETH_DMACCR).

2. Disable the MAC receiver by clearing RE bit of Operating mode configuration register (ETH_MACCR).

3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of Tx queue debug register (ETH_MTLTXQDR) (TRCSTS[1:0] is not 01).
   or
   Flush the Tx FIFO for faster empty operation.

4. Disable the MAC transmitter by clearing TE bit of the Operating mode configuration register (ETH_MACCR) Register.

5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in Tx queue debug register (ETH_MTLTXQDR) and RXQSTS[1:0] to 00 in Rx queue debug register (ETH_MTLRXQDR)).

6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.

7. Restart the operation by starting the Tx DMA. Then enable the MAC Transmitter and Receiver.
   The Rx DMA does not need to be enabled: since the Receiver is disabled, there are no data in the Rx FIFO.
Transmit and Receive clocks are stopped when the link is down

Complete the following steps when the link is down and the Transmit and Receive clocks are stopped:

1. Disable the MAC Transmitter and Receiver by clearing RE and TE bits in the *Operating mode configuration register* (*ETH_MACCR*). This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register* (*ETH_MACDR*) (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC Transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC Transmitter and Receiver by setting RE and TE bits.

### 64.9.9 Programming guidelines for IEEE 1588 timestamping

#### Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control Register* (*ETH_MACTSCR*). However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of *Interrupt enable register* (*ETH_MACIER*).
2. Set bit 0 of *Timestamp control Register* (*ETH_MACTSCR*) to enable timestamping.
3. Program *Subsecond increment register* (*ETH_MACSSIR*) based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register* (*ETH_MACTSAR*) and set bit 5 of *Timestamp control Register* (*ETH_MACTSCR*).
5. Poll the *Timestamp control Register* (*ETH_MACTSCR*) until bit 5 is cleared.
6. Program bit 1 of *Timestamp control Register* (*ETH_MACTSCR*) to select the Fine Update method (if required).
7. Program *System time seconds update register* (*ETH_MACSTSUR*) and *System time nanoseconds update register* (*ETH_MACSTNUR*) with the appropriate time value.
8. Set bit 2 in *Timestamp control Register* (*ETH_MACTSCR*).

The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:

a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

b) Program *Timestamp Ingress asymmetric correction register* (*ETH_MACTSIACR*) to update the correction field in PDelay_Req PTP messages.

9. Enable the MAC receiver and transmitter for proper timestamping.

*Note:* If timestamp operation is disabled by clearing bit 0 of *Timestamp control Register* (*ETH_MACTSCR*), repeat all these steps to restart the timestamp operation.
System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (System time seconds update register (ETH_MACSTSUR) and System time nanoseconds update register (ETH_MACSTNUR)).
2. Set bit 3 (TSUPDT) of the Timestamp control Register (ETH_MACTSCR).
   The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in Section : System time register module, calculate at which rate you intend to increment or decrement the system time.
2. Update the Timestamp addend register (ETH_MACTSAR) with the new value and set bit 5 of the Timestamp control Register (ETH_MACTSCR) Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR).
5. Enable the Timestamp interrupt in bit 12 of Interrupt enable register (ETH_MACIER).
6. Set bit 4 in Register Timestamp control Register (ETH_MACTSCR).
7. When this trigger generates an interrupt, read Interrupt status register (ETH_MACISR).
8. Reprogram Timestamp addend register (ETH_MACTSAR) with the old value and set bit 5 again.

64.9.10 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPTYPSEL, TSMSTRENA, and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCECN bit of PTP Offload control register (ETH_MACPOCR) to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP
5. Program the LSI field of Log message interval register (ETH_MACLMIR) to program the periodicity of the PTP Sync messages.

For example, a value of 1 corresponds to 2^1 which translates to PTP Sync message every 2 seconds, and a value of 0xFF (two's complement of -1) corresponds to 2^-1 which translates to PTP Sync message every 0.536 seconds.

6. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR).

Programming guidelines to enable periodic generation of PTP Pdelay_Req messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.

3. Program the APDREQEN bit of PTP Offload control register (ETH_MACPOCR) to enable periodic generation of PTP Pdelay_Req messages.

4. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to send in egress PTP Pdelay_Req message.

5. Program the LMPDRI field of Log message interval register (ETH_MACLMIR) to program the periodicity of the PTP Pdelay_Req messages.

For example, a value of 1 corresponds to 2^1 which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (two's complement of -1) corresponds to 2^-1 which translates to PTP Pdelay_Req message every 0.536 seconds.

6. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR).

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):
1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 0, 1, and 1 respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.

3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.

4. Program the DRSYNCR and LSI fields in Log message interval register (ETH_MACLMRI). The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 0, 0, and 1 respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.

3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Sync message and send in egress PTP Delay_Req message.

4. Program the DRSYNCR field in Log message interval register (ETH_MACLMRI) to 1 to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 1, 0, and 1 respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.

3. Program the DRSYNCR field in Log message interval register (ETH_MACLMRI) to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync or Pdelay_Req messages.
Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.

4. Program the DRSYNCR and LMPDRI fields in Log message interval register (ETH_MACLMIR) to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.

5. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) for egress PTP Pdelay_Req and Pdelay_Resp messages.

**Programming guidelines to enable the generation of PTP response messages for Transparent Master mode**

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 1, 1, and 1 respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync messages.

3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.

4. Program the DRSYNCR, LSI and LMPDRI fields in Log message interval register (ETH_MACLMIR), the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.

5. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

**Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode**

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):
1. Program the SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of Timestamp control Register (ETH_MACTSCR) to 3, X, and X respectively.

2. Program the PTOEN bit and DN field of PTP Offload control register (ETH_MACPOCR) to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.

3. Program the 80-bit Source Port Identity in PTP Source Port Identity 0 Register (ETH_MACSPI0R), PTP Source port identity 1 register (ETH_MACSPI1R) and PTP Source port identity 2 register (ETH_MACSPI2R) to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.

4. Program the LMPDRI field in Log message interval register (ETH_MACLMIR) to indicate periodicity of the PTP Pdelay_Req messages.

5. Program the TSIE bit of Interrupt enable register (ETH_MACIER) to enable generation of Timestamp interrupt.

6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in Timestamp status register (ETH_MACTSSR). It indicates that the timestamp for PTP Sync message is captured in Tx timestamp status seconds register (ETH_MACTXTSSSR) and Tx timestamp status nanoseconds register (ETH_MACTXTSSNR) for egress PTP Pdelay_Req and Pdelay_Resp messages.

64.9.11 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the LPI control and status register (ETH_MACLCSR) to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.

2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).

3. Program bits 25 to 16 and bits 15 to 0 in LPI timers control register (ETH_MACLTCR).

4. Read the PHY link status by using the MDIO interface and update bit 17 of LPI control and status register (ETH_MACLCSR).

   Update LPI control and status register (ETH_MACLCSR) accordingly. This update should be done whenever the link status in the PHY chip changes.

5. Program One-microsecond-tick counter register (ETH_MAC1USTCR) as per the frequency of the clock used for accessing the CSR slave port.

6. Program the LPIET bit in the LPI entry timer register (ETH_MACLETR) with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bits 20 to 19) of **LPI control and status register (ETH_MACLCSR)** to enable LPI auto-entry and MAC auto-exit from LPI state.

8. Set bit 16 of **LPI control and status register (ETH_MACLCSR)** to put the MAC transmitter in LPI state.

   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.

9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC Transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.

10. The MAC Transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.

11. Reset LPIEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC Transmitter from LPI state.

**Note:** To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in **LPI control and status register (ETH_MACLCSR)**.

   To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of **LPI control and status register (ETH_MACLCSR)** to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

**Gating Off the CSR Clock in the LPI mode**

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

**Gating off the CSR clock in the Rx LPI mode**

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of **LPI control and status register (ETH_MACLCSR)**) is set.

2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the **LPI control and status register (ETH_MACLCSR)**.

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when **LPI control and status register (ETH_MACLCSR)** is read.

**Gating off the CSR clock in the Tx LPI mode**

The following operations are performed when bit 16 (LPIEN) of **LPI control and status register (ETH_MACLCSR)** is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of **LPI control and status register (ETH_MACLCSR)**) is set.

2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the **LPI control and status register (ETH_MACLCSR)**.
After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off, the events on the MAC receiver do not get reported or updated in the CSR. To restore the CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR clock is resumed, reset bit 16 (LPIEN) of the LPI control and status register (ETH_MACLCSR) to exit the MAC from LPI mode.

64.9.12 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in PPS control register (ETH_MACPPSCR). This instructs the MAC to use the Target Time registers (PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)) as start time of PPS signal output.

2. Program the start time value in the Target Time registers (register PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)).

3. Program the width of the PPS signal output in PPS width register (ETH_MACPPSWR) Register.

4. Program PPSCMD[3:0] of PPS control register (ETH_MACPPSCR) to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.

2. Program the width of the next PPS signal output in PPS width register (ETH_MACPPSWR).

3. Program PPSCMD[3:0] bits of PPS control register (ETH_MACPPSCR) to generate a single pulse after the previous pulse is deasserted. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in Target Time registers.

If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the peripheral may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in PPS control register (ETH_MACPPSCR). This instructs the MAC to use the Target Time registers (PPS
target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR) for start time of the PPS signal output.

2. Program the start time value in the Target Time registers (register PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)).

3. Program the interval value between the train of pulses on the PPS signal output in PPS interval register (ETH_MACPPSIR).

4. Program the width of the PPS signal output in PPS width register (ETH_MACPPSWR).

5. Program PPSCMD[3:0] bits in PPS control register (ETH_MACPPSCR) to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a ‘STOP Pulse train at time’ or ‘STOP Pulse Train immediately’ commands.

6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in PPS target time nanoseconds register (ETH_MACPPSTTNR) is reset before programming the Target Time registers again.

7. Program the PPSCMD[3:0] bits in PPS control register (ETH_MACPPSCR) to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

   The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field.

   Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed.

   The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL[1:0] bits in PPS control register (ETH_MACPPSCR) enable you to program the Target Time registers (PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of PPS control register (ETH_MACPPSCR) to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.

2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

Note: The TSTRGTERR0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to
the system time in the System time seconds register (ETH_MACSTSR) and System time nanoseconds register (ETH_MACSTNR).

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register (ETH_MACIER) is set.

Therefore, to avoid unwanted interrupt, the correct writing order is as follow:

1. PPS target time nanoseconds register (ETH_MACPPSTTNR).
2. PPS target time seconds register (ETH_MACPPSTTSR).
3. PPS interval register (ETH_MACPPSIR).
4. PPS width register (ETH_MACPPPSWR).
5. PPSCTRL[3:0] and PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register (ETH_MACPPSCR).

64.9.13 Programming guidelines for TSO

The TCP Segmentation Offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding Channel transmit control register (ETH_DMACTXCR) to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
      – MSS[13:0] of Channel control register (ETH_DMACCR)
      – or MSS in the context descriptor
        If MSS[13:0] field is programmed in both Channel control register (ETH_DMACCR) and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.
### 64.9.14 Programming guidelines to perform VLAN filtering on the receiver

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program **VLAN tag register (ETH_MACVTR)** for the following bit to select the filtering method:
   - **ETV**: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   - **VTHM**: VLAN Tag Hash Table Match Enable.
   - **ERIVLT**: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should be enabled by setting EDVLP)
   - **ERSVLM**: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   - **DOVLTC**: Ignores VLAN Type for Tag Match
   - **VTIM**: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching

2. Program **VL** bit in **VLAN tag register (ETH_MACVTR)** for the 12-bit or 16-bit VLAN tag.

3. If VLAN tag Hash filtering is enabled, program **VLAN Hash table register (ETH_MACVHTR)**:
   - When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the **VLAN Hash table register (ETH_MACVHTR)**.
   - When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of **VLAN Hash table register (ETH_MACVHTR)**.

For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

### 64.10 Descriptors

#### 64.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**
  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

- **Context descriptors**
  The context descriptors are used to provide control information applicable to the packet to be transmitted.

Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.
64.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 978. Descriptor ring structure

In a ring structure, descriptors are separated by the 32-bit word number programmed in the DSL field of the Channel control register (ETH_DMACCR). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- Channel Tx descriptor ring length register (ETH_DMACTXRLR)
- Channel Rx descriptor ring length register (ETH_DMACRXRLR)

The Channel Tx descriptor tail pointer register (ETH_DMACTXDTPR) or Channel Rx descriptor tail pointer register (ETH_DMACRXDTPR) contains the pointer to the descriptor address (N). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (N – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

Current Descriptor Pointer == Descriptor Tail Pointer;

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the descriptor tail pointer register and update the tail pointer so that the following condition is met:

Descriptor Tail Pointer > Current Descriptor Pointer;
The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 979: DMA descriptor ring*.

For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.
Descriptor tail pointer handling examples

Figure 980. Descriptor tail pointer example 1

When the current descriptor pointer points to descriptor 4 (D4) and the descriptor tail pointer points to descriptor 7 (D7), the last descriptor owned by the DMA is descriptor 6 (D6) and there are three descriptors (D4, D5, D6) available for the DMA, as shown in Figure 980: Descriptor tail pointer example 1.

Figure 981. Descriptor tail pointer example 2

As shown in Figure 981: Descriptor tail pointer example 2, the application frees four descriptors (D7, D8, D9 and D0) and updates the descriptor tail pointer to point to descriptor 1 (D1).
The last descriptor owned by the DMA is descriptor 0 (D0). The descriptors referenced between the current descriptor pointer (D4) and the descriptor tail pointer (D1) are available to the DMA.

64.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format.

Transmit normal descriptor (read format)

Figure 982 shows the Read format for Transmit normal descriptor. Table 725 to Table 728 provide a detailed description of all Transmit normal descriptors (read format).

Figure 982. Transmit descriptor (read format)

- TDES0 normal descriptor (read format)

Table 725. TDES0 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF1AP</td>
<td>Buffer 1 Address Pointer or TSO Header Address Pointer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– TSE bit of TDES3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– FD bit of TDES3</td>
</tr>
</tbody>
</table>
- **TDES1 normal descriptor (read format)**

**Table 726. TDES1 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF2AP</td>
<td><strong>Buffer 2 or Buffer 1 Address Pointer:</strong> These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment.</td>
</tr>
</tbody>
</table>

- **TDES2 normal descriptor (read format)**

**Table 727. TDES2 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>IOC</td>
<td><strong>Interrupt on completion:</strong> This bit sets the TI bit in the Channel status register (ETH_DMACSR) when the present packet transmission is complete.</td>
</tr>
<tr>
<td>30</td>
<td>TTSE</td>
<td><strong>Transmit Timestamp Enable</strong> This bit enables the IEEE1588 timestamping for Transmit packet referenced by the descriptor.</td>
</tr>
<tr>
<td>29:16</td>
<td>B2L</td>
<td><strong>Buffer 2 Length</strong> The driver sets this field. When set, this field indicates Buffer 2 length.</td>
</tr>
</tbody>
</table>
| 15:14  | VTIR   | **VLAN Tag Insertion or Replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows:
00: Do not add a VLAN tag.
01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets.
10: Insert a VLAN tag with the tag value programmed in the VLAN inclusion register (ETH_MACVIR) or context descriptor.
11: Replace the VLAN tag in packets with the tag value programmed in the VLAN inclusion register (ETH_MACVIR) or context descriptor. This option should be used only with the VLAN packets. |
Ethernet (ETH): media access control (MAC) with DMA controller

- TDES3 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 13:0 | HL or B1L | Header length or buffer 1 length  
For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length.  
If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes.  
If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length. |

Table 727. TDES2 normal descriptor (read format) (continued)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31   | OWN  | Own bit  
1: the DMA owns the descriptor.  
0: the application owns the descriptor.  
The DMA clears this bit after it completes the transfer of data given in the associated buffer(s). |
| 30   | CTXT | Context Type  
This bit should be set to 0 for normal descriptor. |
| 29   | FD   | First Descriptor  
When this bit is set, it indicates that the buffer contains the first segment of a packet. |
| 28   | LD   | Last Descriptor  
When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value. |

Table 728. TDES3 normal descriptor (read format)

<table>
<thead>
<tr>
<th>Bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31   | OWN  | Own bit  
1: the DMA owns the descriptor.  
0: the application owns the descriptor.  
The DMA clears this bit after it completes the transfer of data given in the associated buffer(s). |
| 30   | CTXT | Context Type  
This bit should be set to 0 for normal descriptor. |
| 29   | FD   | First Descriptor  
When this bit is set, it indicates that the buffer contains the first segment of a packet. |
| 28   | LD   | Last Descriptor  
When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value. |
This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following:

**00**: CRC and Pad Insertion

The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes.

**01**: CRC Insertion (Disable Pad Insertion)

The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes.

**10**: Disable CRC Insertion

The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.

**11**: CRC Replacement

The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer.

When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.

These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet.

Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement.

The following list describes the values of Bits[24:23]:

**00**: Do not include the source address

**01**: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses.

**10**: Replace the source address. For reliable transmission, the application must provide frames with source addresses.

**11**: Reserved

These bits are valid when the First Segment control bit (TDES3[29]) is set.

If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor.

When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set.
Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

*Figure 983* shows the write-back format for Transmit normal descriptors. *Table 729 to Table 732* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).
Figure 983. Transmit descriptor write-back format

- **TDES0 normal descriptor (write-back format)**

  **Table 729. TDES0 normal descriptor (write-back format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0| TTSL  | Transmit Packet Timestamp Low  
  The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.  
  1. This format is only applicable to the last descriptor of a packet. |

- **TDES1 normal descriptor (write-back format)**

  **Table 730. TDES1 normal descriptor (write-back format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0| TTSH  | Transmit Packet Timestamp High  
  The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding Receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.  
  1. This format is only applicable to the last descriptor of a packet. |

- **TDES2 normal descriptor (write-back format)**
- **TDES3 normal descriptor (write-back format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31  | OWN  | **Own bit**
When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0. |
| 30  | CTXT | **Context Type**
This bit should be set to 0 for normal descriptors. |
| 29  | FD   | **First Descriptor**
This bit indicates that the buffer contains the first segment of a packet. |
| 28  | LD   | **Last Descriptor**
This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet. |
| 27:18 | | **Reserved** |
| 17  | TTSS | **Tx Timestamp Status**
This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set. |
| 16  | | **Reserved** |
### Error Summary
This bit indicates the logical OR of the following bits:
- TDES3[0]: IP Header Error
- TDES3[14]: Jabber Timeout
- TDES3[13]: Packet Flush
- TDES3[12]: Payload Checksum Error
- TDES3[11]: Loss of Carrier
- TDES3[10]: No Carrier
- TDES3[9]: Late Collision
- TDES3[8]: Excessive Collision
- TDES3[3]: Excessive Deferral
- TDES3[2]: Underflow Error

### Jabber Timeout
This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the Operating mode configuration register (ETH_MACCR) is not set.

### Packet Flushed
This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.

### Payload Checksum Error
This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.

### Loss of Carrier
This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.

### No Carrier
This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.

### Late Collision
This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>ES</td>
<td>Error Summary&lt;br&gt;This bit indicates the logical OR of the following bits:&lt;br&gt;- TDES3[0]: IP Header Error&lt;br&gt;- TDES3[14]: Jabber Timeout&lt;br&gt;- TDES3[13]: Packet Flush&lt;br&gt;- TDES3[12]: Payload Checksum Error&lt;br&gt;- TDES3[11]: Loss of Carrier&lt;br&gt;- TDES3[10]: No Carrier&lt;br&gt;- TDES3[9]: Late Collision&lt;br&gt;- TDES3[8]: Excessive Collision&lt;br&gt;- TDES3[3]: Excessive Deferral&lt;br&gt;- TDES3[2]: Underflow Error</td>
</tr>
<tr>
<td>14</td>
<td>JT</td>
<td>Jabber Timeout&lt;br&gt;This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the Operating mode configuration register (ETH_MACCR) is not set.</td>
</tr>
<tr>
<td>13</td>
<td>FF</td>
<td>Packet Flushed&lt;br&gt;This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.</td>
</tr>
<tr>
<td>12</td>
<td>PCE</td>
<td>Payload Checksum Error&lt;br&gt;This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.</td>
</tr>
<tr>
<td>11</td>
<td>LoC</td>
<td>Loss of Carrier&lt;br&gt;This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.</td>
</tr>
<tr>
<td>10</td>
<td>NC</td>
<td>No Carrier&lt;br&gt;This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.</td>
</tr>
<tr>
<td>9</td>
<td>LC</td>
<td>Late Collision&lt;br&gt;This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode). This bit is not valid if Underflow Error is set.</td>
</tr>
</tbody>
</table>
Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag.

---

Table 732. TDES3 normal descriptor (write-back format)(1) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 8   | EC   | Excessive Collision  
This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the Operating mode configuration register (ETH_MACCR), this bit is set after first collision and the transmission of the packet is aborted. |
| 7:4 | CC   | Collision Count  
This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set. |
| 3   | ED   | Excessive Deferral  
This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times if DC bit is set in the Operating mode configuration register (ETH_MACCR). |
| 2   | UF   | Underflow Error  
This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions:  
The DMA encountered an empty Transmit Buffer while transmitting the packet  
The application filled the MTL Tx FIFO slower than the MAC transmit rate  
The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register. |
| 1   | DB   | Deferred Bit  
This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode. |
| 0   | IHE  | IP Header Error  
When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload. |

1. This format is only applicable to the last descriptor of a packet.
stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN Tag control input.

Figure 984 shows the format for Transmit context descriptors. Table 733 to Table 736 provide a detailed description of all Transmit context descriptors.

**Figure 984. Transmit context descriptor format**

![Transmit context descriptor format diagram]

- TDES0 context descriptor (read format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>TTSL</td>
<td>Transmit Packet Timestamp Low</td>
</tr>
</tbody>
</table>
|     |               | For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)
**Table 734. TDES1 context descriptor**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:0 | TTSH | Transmit Packet Timestamp High  
For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set. |

**Table 735. TDES2 context descriptor**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:16 | IVT  | Inner VLAN Tag  
When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets. |
| 15:14 | Reserved |
| 13:0 | MSS  | Maximum Segment Size  
This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset. |

**Table 736. TDES3 context descriptor**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31 | OWN | Own bit  
1: the DMA owns the descriptor.  
0: the application owns the descriptor.  
The DMA clears this bit immediately after a read operation. |
| 30 | CTXT | Context Type  
This bit should be set to 1 for context descriptor. |
| 29:28 | Reserved |
| 27 | OSTC | One-Step Timestamp Correction Enable  
When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1. |
One-Step Timestamp Correction Input or MSS Valid
When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.

Context Descriptor Error
When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor.

Context Descriptor errors can be:
- Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet.
- All 1s.

Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel status register (ETH_DMACSR).

Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent. This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported).

Inner VLAN Tag Insert or Replace
When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes.

This bitfield has the following values:
00: Do not add the inner VLAN tag.
01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames.
10: Insert an inner VLAN tag with the tag value programmed in the Inner VLAN inclusion register (ETH_MACIVIR) or context descriptor.
11: Replace the inner VLAN tag in packets with the tag value programmed in the Inner VLAN inclusion register (ETH_MACIVIR) or context descriptor. This option should be used only with the VLAN frames.

Inner VLAN Tag Valid
When this bit is set, it indicates that the IVT field of TDES2 is valid.
Table 736. TDES3 context descriptor (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 16   | VLTV | VLAN Tag Valid  
When this bit is set, it indicates that the VT field of TDES3 is valid. |
| 15:0 | VT   | VLAN Tag  
This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the VLAN inclusion register (ETH_MACVIR) is reset. |
64.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following Receive descriptors are present:
- Normal descriptors with read and write-back formats
- Context descriptors

All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see Figure 985: Receive normal descriptor (read format) for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in Figure 986: Receive normal descriptor (write-back format).

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in Figure 987: Receive context descriptor.

Receive normal descriptor (read format)

Figure 985 shows the read format for Receive normal descriptors. Table 737 to Table 740 provide a detailed description of all Receive normal descriptors (read format).

![Figure 985. Receive normal descriptor (read format)](image)

**Note:** In the Receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.
- RDES0 normal descriptor (read format)

**Table 737. RDES0 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF1AP</td>
<td><strong>Buffer 1 Address Pointer</strong>&lt;br&gt;These bits indicate the physical address of Buffer 1.&lt;br&gt;The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES2[1:0]=0 (in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.</td>
</tr>
</tbody>
</table>

- RDES1 normal descriptor (read format)

**Table 738. RDES1 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- RDES2 normal descriptor (read format)

**Table 739. RDES2 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>BUF2AP</td>
<td><strong>Buffer 2 Address Pointer</strong>&lt;br&gt;These bits indicate Buffer 2 physical address.&lt;br&gt;The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[1:0]=0 and writes to the complete location.</td>
</tr>
</tbody>
</table>

- RDES3 normal descriptor (read format)

**Table 740. RDES3 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OWN</td>
<td><strong>Own bit</strong>&lt;br&gt;When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true:&lt;br&gt;– The DMA completes the packet reception&lt;br&gt;– The buffers associated with the descriptor are full</td>
</tr>
</tbody>
</table>
Receive normal descriptor (write-back format)

*Figure 986* shows the write-back format for Receive normal descriptors. *Table 741* to *Table 744* provide a detailed description of all Receive normal descriptors (write-back format).

---

**Table 740. RDES3 normal descriptor (read format)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>IOC</td>
<td><strong>Interrupt Enabled on Completion</strong>&lt;br&gt;When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.</td>
</tr>
<tr>
<td>29:26</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>BUF2V</td>
<td><strong>Buffer 2 Address Valid</strong>&lt;br&gt;When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.</td>
</tr>
<tr>
<td>24</td>
<td>BUF1V</td>
<td><strong>Buffer 1 Address Valid</strong>&lt;br&gt;When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid.&lt;br&gt;The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.</td>
</tr>
<tr>
<td>23:0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>

**Receive normal descriptor (write-back format)**

*Figure 986* shows the write-back format for Receive normal descriptors. *Table 741* to *Table 744* provide a detailed description of all Receive normal descriptors (write-back format).
• RDES0 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:16</td>
<td>IVT</td>
<td>Inner VLAN Tag&lt;br&gt;This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.</td>
</tr>
<tr>
<td>15:0</td>
<td>OVT</td>
<td>Outer VLAN Tag&lt;br&gt;This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.</td>
</tr>
</tbody>
</table>

Table 741. RDES0 normal descriptor (write-back format)

• RDES1 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPC</td>
<td>OAM Subtype Code, or MAC Control Packet opcode&lt;br&gt;OAM Subtype Code&lt;br&gt;If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields.&lt;br&gt;MAC Control Packet opcode&lt;br&gt;If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.</td>
<td></td>
</tr>
<tr>
<td>TD</td>
<td>Timestamp Dropped&lt;br&gt;This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.</td>
<td></td>
</tr>
<tr>
<td>TSA</td>
<td>Timestamp Available&lt;br&gt;When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3[28]) is set.&lt;br&gt;The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.</td>
<td></td>
</tr>
<tr>
<td>PV</td>
<td>PTP Version&lt;br&gt;1: Received PTP message in IEEE 1588 version 2 format&lt;br&gt;0: Received PTP message in IEEE 1588 version 1 format</td>
<td></td>
</tr>
<tr>
<td>PFT</td>
<td>PTP Packet Type&lt;br&gt;This bit indicates that the PTP message is sent directly over Ethernet.</td>
<td></td>
</tr>
</tbody>
</table>

Table 742. RDES1 normal descriptor (write-back format)(1)
### Table 742. RDES1 normal descriptor (write-back format)\(^{(1)}\) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:8</td>
<td>PMT</td>
<td><strong>PTP Message Type</strong>&lt;br&gt;These bits are encoded to give the type of the message received:&lt;br&gt;0000: No PTP message received&lt;br&gt;0001: SYNC (all clock types)&lt;br&gt;0010: Follow_Up (all clock types)&lt;br&gt;0011: Delay_Req (all clock types)&lt;br&gt;0100: Delay_Resp (all clock types)&lt;br&gt;0101: Pdelay_Req (in peer-to-peer transparent clock)&lt;br&gt;0110: Pdelay_Resp (in peer-to-peer transparent clock)&lt;br&gt;0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock)&lt;br&gt;1000: Announce&lt;br&gt;1001: Management&lt;br&gt;1010: Signaling&lt;br&gt;1011–1110: Reserved&lt;br&gt;1111: PTP packet with Reserved message type</td>
</tr>
<tr>
<td>7</td>
<td>IPCE</td>
<td><strong>IP Payload Error</strong>&lt;br&gt;When this bit is set, it indicates either of the following:&lt;br&gt;– The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment.&lt;br&gt;– The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field.&lt;br&gt;– The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP.&lt;br&gt;Bit 15 (ES) of RDES3 is not set when this bit is set.</td>
</tr>
<tr>
<td>6</td>
<td>IPCB</td>
<td><strong>IP Checksum Bypassed</strong>&lt;br&gt;This bit indicates that the checksum offload engine is bypassed.</td>
</tr>
<tr>
<td>5</td>
<td>IPV6</td>
<td><strong>IPv6 header Present</strong>&lt;br&gt;This bit indicates that an IPv6 header is detected.</td>
</tr>
<tr>
<td>4</td>
<td>IPV4</td>
<td><strong>IPv4 Header Present</strong>&lt;br&gt;This bit indicates that an IPv4 header is detected.</td>
</tr>
<tr>
<td>3</td>
<td>IPHE</td>
<td><strong>IP Header Error</strong>&lt;br&gt;– When this bit is set, it indicates either of the following:&lt;br&gt;– The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes.&lt;br&gt;– The IP datagram version is not consistent with the Ethernet Type value.&lt;br&gt;– Ethernet packet does not have the expected number of IP header bytes.&lt;br&gt;This bit is valid when either bit 5 or bit 4 is set.</td>
</tr>
</tbody>
</table>
### Table 742. RDES1 normal descriptor (write-back format) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2:0 | PT   | Payload Type  
These bits indicate the type of payload encapsulated in the IP datagram processed by the Receive Checksum Offload Engine (COE):  
- 000: Unknown type or IP/AV payload not processed  
- 001: UDP  
- 010: TCP  
- 011: ICMP  
- 100: IGMP if IPV4 Header Present bit is set  
Others: reserved.  
If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000. |

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

### RDES2 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3L4FM</td>
<td>L4FM</td>
<td>L3FM</td>
<td>MADRM</td>
<td>HF</td>
<td>DAF</td>
<td>SAF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>VF</td>
<td>Reserved</td>
<td>ARPRN</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 743. RDES2 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 31:29 | L3L4FM | Layer 3 and Layer 4 Filter Number Matched  
These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet:  
- 000: Filter 0  
- 001: Filter 1  
- 010: Filter 2  
- 011: Filter 3  
- 100: Filter 4  
- 101: Filter 5  
- 110: Filter 6  
- 111: Filter 7  
This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter. |

| 28 | L4FM | Layer 4 Filter Match  
When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true:  
- Layer 3 fields are not enabled and all enabled Layer 4 fields match  
- All enabled Layer 3 and Layer 4 filter fields match  
When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29]. |
**Table 743. RDES2 normal descriptor (write-back format) (continued)**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 27  | L3FM | Layer 3 Filter Match  
When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true:  
– All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed  
– All enabled filter fields match  
When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29]. |
| 26:19 | MADRM | MAC Address Match or Hash Value  
When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset.  
When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register. |
| 18  | HF   | Hash Filter Status  
When this bit is set, it indicates that the packet passed the MAC address Hash filter. Its[26:19] indicate the Hash value. |
| 17  | DAF  | Destination Address Filter Fail  
When this bit is set, it indicates that the packet failed the DA Filter in the MAC. |
| 16  | SAF  | SA Address Filter Fail  
When this bit is set, it indicates that the packet failed the SA Filter in the MAC. |
| 15  | VF   | VLAN Filter Status  
When this bit is set, it indicates that the VLAN Tag of received packet passed the VLAN filter. |
| 14:11 | Reserved |  |
| 10  | ARPNR | ARP Reply Not Generated  
When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time). |
| 9:0 | Reserved |  |

- RDES3 normal descriptor (write-back format)

```
+----------------+----------------+
<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
+----------------+----------------+
| 31  | OWN  |
| 30  | CTXT |
| 29  | FD   |
| 28  | LD   |
| 27  | RS2V |
| 26  | RS1V |
| 25  | RS0V |
| 24  | CE   |
| 23  | GP   |
| 22  | RWT  |
| 21  | OE   |
| 20  | RE   |
| 19  | DE   |
| 18  | LT   |
| 17  |        |             |
| 16  |        |             |
| 15  |        |             |
| 14  |        |             |
| 13  |        |             |
| 12  |        |             |
| 11  |        |             |
| 10  |        |             |
| 9   |        |             |
| 8   |        |             |
| 7   |        |             |
| 6   |        |             |
| 5   |        |             |
| 4   |        |             |
| 3   |        |             |
| 2   |        |             |
| 1   |        |             |
| 0   |        |             |
+----------------+----------------+            
| ES | PL   |
+----------------+----------------+
```
Table 744. RDES3 normal descriptor (write-back format)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OWN</td>
<td>Own bit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: The DMA owns the descriptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: The application owns the descriptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The DMA clears this bit when either of the following conditions is true:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The DMA completes the packet reception</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The buffers associated with the descriptor are full</td>
</tr>
<tr>
<td>30</td>
<td>CTXT</td>
<td>Receive Context Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When CTXT and FD bits are used together, {CTXT, FD} possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00: Intermediate Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01: First Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10: Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11: Descriptor error (due to all 1s)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: When a Descriptor error occurs, the Receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel status register (ETH_DMACSRR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.</td>
</tr>
<tr>
<td>29</td>
<td>FD</td>
<td>First Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.</td>
</tr>
<tr>
<td>28</td>
<td>LD</td>
<td>Last Descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.</td>
</tr>
<tr>
<td>27</td>
<td>RS2V</td>
<td>Receive Status RDES2 Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>26</td>
<td>RS1V</td>
<td>Receive Status RDES1 Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>25</td>
<td>RS0V</td>
<td>Receive Status RDES0 Valid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.</td>
</tr>
</tbody>
</table>
### Table 744. RDES3 normal descriptor (write-back format) (continued)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>CE</td>
<td><strong>CRC Error</strong>&lt;br&gt;When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.</td>
</tr>
<tr>
<td>23</td>
<td>GP</td>
<td><strong>Giant Packet</strong>&lt;br&gt;When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set).&lt;br&gt;Giant packet indicates only the packet length. It does not cause any packet truncation.</td>
</tr>
<tr>
<td>22</td>
<td>RWT</td>
<td><strong>Receive Watchdog Timeout</strong>&lt;br&gt;When this bit is set, it indicates that the Receive Watchdog Timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.</td>
</tr>
<tr>
<td>21</td>
<td>OE</td>
<td><strong>Overflow Error</strong>&lt;br&gt;When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO.&lt;br&gt;This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.</td>
</tr>
<tr>
<td>20</td>
<td>RE</td>
<td><strong>Receive Error</strong>&lt;br&gt;When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception.</td>
</tr>
<tr>
<td>19</td>
<td>DE</td>
<td><strong>Dribble Bit Error</strong>&lt;br&gt;When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.</td>
</tr>
</tbody>
</table>
| 18:16 | LT   | **Length/Type Field**<br>This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows:<br>000: The packet is a length packet<br>001: The packet is a type packet.<br>011: The packet is a ARP Request packet type<br>100: The packet is a type packet with VLAN Tag<br>101: The packet is a type packet with Double VLAN tag<br>110: The packet is a MAC Control packet type<br>111: The packet is a OAM packet type<br>010: Reserved
### Error Summary

When this bit is set, it indicates the logical OR of the following bits:
- RDES3[19]: Dribble Error
- RDES3[20]: Receive Error
- RDES3[21]: Overflow Error
- RDES3[22]: Watchdog Timeout
- RDES3[23]: Giant Packet
- RDES3[24]: CRC Error

This field is valid only when the LD bit of RDES3 is set.

### Packet Length

These bits indicate the byte length of the received packet that was transferred to system memory (including CRC).

This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet.

When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 15  | ES   | Error Summary  
When this bit is set, it indicates the logical OR of the following bits:  
RDES3[19]: Dribble Error  
RDES3[20]: Receive Error  
RDES3[21]: Overflow Error  
RDES3[22]: Watchdog Timeout  
RDES3[23]: Giant Packet  
RDES3[24]: CRC Error  
This field is valid only when the LD bit of RDES3 is set. |
| 14:0| PL   | Packet Length  
These bits indicate the byte length of the received packet that was transferred to system memory (including CRC).  
This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet.  
When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet. |
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 987 shows the format for Receive context descriptors. Table 745 to Table 748 provide a detailed description of all Receive context descriptors.

Figure 987. Receive context descriptor

- RDES0 context descriptor

Table 745. RDES0 context descriptor

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RTSL</td>
<td>Receive Packet Timestamp Low&lt;br&gt;The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding Receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.</td>
</tr>
</tbody>
</table>
- RDES1 context descriptor

**Table 746. RDES1 context descriptor**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>RTSH</td>
<td><strong>Receive Packet Timestamp High</strong>&lt;br&gt;The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.</td>
</tr>
</tbody>
</table>

- RDES2 context descriptor

**Table 747. RDES2 context descriptor**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

- RDES3 context descriptor

**Table 748. RDES3 context descriptor**

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>OWN</td>
<td><strong>Own Bit</strong>&lt;br&gt;1: The DMA owns the descriptor&lt;br&gt;0: The application owns the descriptor.&lt;br&gt;The DMA clears this bit when either of the following conditions is true:&lt;br&gt;The DMA completes the packet reception&lt;br&gt;The buffers associated with the descriptor are full</td>
</tr>
<tr>
<td>30</td>
<td>CTXT</td>
<td><strong>Receive Context Descriptor</strong>&lt;br&gt;When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor.</td>
</tr>
<tr>
<td>29:0</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
64.11 Ethernet registers

64.11.1 Ethernet register maps

This section provides the following register maps:
- DMA registers (see Section 64.11.2: Ethernet DMA registers)
- MTL registers (see Section 64.11.3: Ethernet MTL registers)
- MAC registers including (see Section 64.11.4: Ethernet MAC and MMC registers)
  - MMC registers

64.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000
Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-18</td>
<td>Reserved</td>
<td>Must be kept at reset value.</td>
</tr>
<tr>
<td>17-16</td>
<td>INTM[1:0]</td>
<td>Interrupt Mode</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field defines the interrupt mode of the Ethernet peripheral.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to Table 724: Transfer complete interrupt behavior).</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14-12</td>
<td>PR[2:0]</td>
<td>Priority ratio</td>
</tr>
<tr>
<td></td>
<td></td>
<td>These bits control the priority ratio in weighted round-robin arbitration between the Rx DMA and Tx DMA. These bits are valid only when the DA bit is reset. The priority ratio is Rx:Tx or Tx:Rx depending on whether the TXPR bit is reset or set.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000: The priority ratio is 1:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: The priority ratio is 2:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: The priority ratio is 3:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: The priority ratio is 4:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: The priority ratio is 5:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: The priority ratio is 6:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110: The priority ratio is 7:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111: The priority ratio is 8:1</td>
</tr>
<tr>
<td>11</td>
<td>TXPR</td>
<td>Transmit priority</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.</td>
</tr>
</tbody>
</table>
System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0000 0000

The System bus mode register controls the behavior of the AHB master. It mainly controls burst splitting and number of outstanding requests.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>RB</td>
<td>MB</td>
<td>AAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>FB</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **RB**: Rebuild INCRx Burst

When this bit is set high and the AHB master gets SPLIT, RETRY, or Early Burst Termination (EBT) response, the AHB master interface rebuilds the pending beats of any initiated burst transfer with INCRx and SINGLE transfers. By default, the AHB master interface rebuilds pending beats of an EBT with an unspecified (INCR) burst.

Bit 14 **MB**: Mixed Burst

When this bit is set high and the FB bit is low, the AHB master performs undefined bursts transfers (INCR) for burst length of 16 or more. For burst length of 16 or less, the AHB master performs fixed burst transfers (INCRx and SINGLE).
Bit 12 **AAL**: Address-Aligned Beats  
When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bits 11:1 Reserved, must be kept at reset value.

Bit 0 **FB**: Fixed Burst Length  
When this bit is set to 1, the AHB master initiates burst transfers of specified length (INCRx or SINGLE).  
When this bit is set to 0, the AHB master initiates transfers of unspecified length (INCR) or SINGLE transfers.
Interrupt status register (ETH_DMAISR)

Address offset: 0x1008
Reset value: 0x0000 0000

The application reads this Interrupt Status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MACIS</td>
<td>MTLIS</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 17 **MACIS**: MAC Interrupt Status
This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL Interrupt Status
This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **DC0IS**: DMA Channel Interrupt Status
This bit indicates an interrupt event in DMA Channel. To reset this bit to 0, the software must read the corresponding register in DMA Channel to get the exact cause of the interrupt and clear its source.

Debug status register (ETH_DMADSR)

Address offset: 0x100C
Reset value: 0x0000 0000

The Debug status register gives the Receive and Transmit process status for DMA Channel for debugging purpose.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RM0477 Ethernet (ETH): media access control (MAC) with DMA controller

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:12 **TPSO[3:0]**: DMA Channel Transmit Process State
This field indicates the Tx DMA FSM state for Channel:
- 000: Stopped (Reset or Stop Transmit Command issued)
- 001: Running (Fetching Tx Transfer Descriptor)
- 010: Running (Waiting for status)
- 011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
- 100: Timestamp write state
- 101: Reserved for future use
- 110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
- 111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPSO[3:0]**: DMA Channel Receive Process State
This field indicates the Rx DMA FSM state for Channel:
- 000: Stopped (Reset or Stop Receive Command issued)
- 001: Running (Fetching Rx Transfer Descriptor)
- 010: Reserved for future use
- 011: Running (Waiting for Rx packet)
- 100: Suspended (Rx Descriptor Unavailable)
- 101: Running (Closing the Rx Descriptor)
- 110: Timestamp write state
- 111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:1  Reserved, must be kept at reset value.

Bit 0 **AXWHSTS**: AHB Master Write Channel
When high, this bit indicates that the write channel of the AHB master FMSs are in non-idle state.

**Channel control register (ETH_DMACCR)**

Address offset: 0x1100

Reset value: 0x0000 0000

The DMA Channel control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.
Bits 31:21  Reserved, must be kept at reset value.

Bits 20:18  **DSL[2:0]:** Descriptor Skip Length  
This bit specifies the 32-bit word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.  
When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17  Reserved, must be kept at reset value.

Bit 16  **PBLX8:** 8xPBL mode  
When this bit is set, the PBL value programmed in Bits[21:16] in *Channel transmit control register (ETH_DMACTXCR)* and in Bits[21:16] in *Channel receive control register (ETH_DMACRXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

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

Bits 13:0  **MSS[13:0]:** Maximum Segment Size  
This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel transmit control register (ETH_DMACTXCR)* is set.  
The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

---

**Channel transmit control register (ETH_DMACTXCR)**

Address offset: 0x1104  
Reset value: 0x0000 0000  
The DMA Channel Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

| Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9  | Bit 8  | Bit 7  | Bit 6  | Bit 5  | Bit 4  | Bit 3  | Bit 2  | Bit 1  | Bit 0  |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     | rw     |
Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **TXPBL[5:0]: Transmit Programmable Burst Length**
These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

- a) Set the PBLx8 mode in ETH_DMACCR.
- b) Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue operating mode register (ETH_MTLTXQOMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 2048 bytes is 512. TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TSE: TCP Segmentation Enabled**
When this bit is set, the DMA performs the TCP segmentation for packets in Channel x. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF: Operate on Second Packet**
When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ST: Start or Stop Transmission Command**
When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted. The DMA tries to acquire descriptor from either of the following positions:

- The current position in the list: this is the base address of the Transmit list set by the ETH_DMACTXDCLR register.
- The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACTXDCLR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACTXDCLR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

**Channel receive control register (ETH_DMACRXCR)**

Address offset: 0x1108
Reset value: 0x0000 0000
The DMA Channel Receive Control register controls the Rx features such as PBL, buffer size, and extended status.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>RPF</td>
<td>DMA Rx Channel Packet Flush</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set to 1, the Ethernet peripheral automatically flushes the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>packet from the Rx queues destined to DMA Rx Channel when the DMA Rx Channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>is stopped after a system bus error has occurred. When this bit remains set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and the DMA is re-started by the software driver, the packets residing in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rx Queues that were received when this RxDMA was stopped, are flushed out.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The packets that are received by the MAC after the RxDMA is re-started are</td>
</tr>
<tr>
<td></td>
<td></td>
<td>routed to the RxDMA. The flushing happens on the Read side of the Rx queue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set to 0 the Ethernet peripheral does not flush the packet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>in the Rx queue destined to DMA Rx Channel after the DMA is stopped due to a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>system bus error. This might cause head-of-line blocking in the corresponding</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RxQueue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: The stopping of packet flow from a Rx DMA Channel to the application by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>channels. In Dynamic mapping mode, setting RPF bit in ETH_DMACRXCR register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>might flush packets from unintended Rx Queues which are destined to the stopped</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rx DMA Channel.</td>
</tr>
</tbody>
</table>

| Bits 30:22 | Reserved, must be kept at reset value. |

| Bits 21:16 | RXPBL[5:0]: Receive Programmable Burst Length |
|           | These bits indicate the maximum number of beats to be transferred in one DMA data transfer. |
|           | This is the maximum value that is used in a single block Read or Write. The DMA always |
|           | attempts to burst as specified in PBL each time it starts a burst transfer on the application |
|           | bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other |
|           | value results in undefined behavior. |
|           | To transfer more than 32 beats, perform the following steps: |
|           | a) Set the PBLx8 mode in the ETH_DMACCR. |
|           | b) Set the RXPBL[5:0]. |
|           | Note: The maximum value of RXPBL must be less than or equal to half the Rx Queue size |
|           | (RQS field of Rx queue operating mode register (ETH_MTLRXQOMR)) in terms of beats. This is |
|           | required so that the Rx Queue has space to store at least another Rx PBL worth of data while |
|           | the MTL Rx Queue Controller is transferring data to MAC. The total locations in Rx Queue |
|           | of size 2048 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to |
|           | 512/2. |
Bit 15  Reserved, must be kept at reset value.

Bits 14:1  **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

**Note:** The buffer size must be a multiple of 4. This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 4, it may result into an undefined behavior.

The LSB bits (1:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0  **SR**: Start or Stop Receive

When this bit is set, the DMA tries to acquire the descriptor from the Receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

- The current position in the list: this is the address set by the Channel Rx descriptor list address register (ETH_DMACRXDLAR).
- The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMACS is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the Channel Rx descriptor list address register (ETH_DMACRXDLAR), the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the Receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

**Channel Tx descriptor list address register (ETH_DMACTXDLAR)**

Address offset: 0x1114

Reset value: 0x0000 0000

Channel Tx Descriptor List Address register points the DMA to the start of Transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in ETH_DMACTXCR register. When stopped, this register can be written with a new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier.
Bits 31:0  **TDESLA[31:0]**: Start of Transmit List
This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel Rx descriptor list address register (ETH_DMACRXDLAR)

Address offset: 0x111C
Reset value: 0x0000 0000
The Channel Rx Descriptor List Address register points the DMA to the start of Receive descriptor list.

This register points to the start of the Receive Descriptor List. The descriptor lists reside in the physical memory space of the application and must be word-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMACRXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDESLA[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **RDESLA[31:0]**: Start of Receive List
This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (1:0) for 32-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).
Channel Tx descriptor tail pointer register (ETH_DM ACTXDTPR)

Address offset: 0x1120
Reset value: 0x0000 0000

The Channel Tx Descriptor Tail Pointer register points to an offset from the base and indicates the location of the last valid descriptor.

Bits 31:0 **TDT[31:0]**: Transmit Descriptor Tail Pointer
This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

Channel Rx descriptor tail pointer register (ETH_DMA CRXDTPR)

Address offset: 0x1128
Reset value: 0x0000 0000

The Channel Rx Descriptor Tail Pointer Points to an offset from the base and indicates the location of the last valid descriptor.

Bits 31:0 **RDT[31:0]**: Receive Descriptor Tail Pointer
This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel Tx descriptor ring length register (ETH_DMACTXRLR)

Address offset: 0x112C
Reset value: 0x0000 0000

The Tx Descriptor Ring Length register contains the length of the Transmit descriptor ring.
**Channel Rx descriptor ring length register (ETH_DMACRXRLR)**

Address offset: 0x1130

Reset value: 0x0000 0000

The Channel Rx Descriptor Ring Length register contains the length of the Receive descriptor circular ring.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

Bits 9:0  **TDRL[9:0]:** Transmit Descriptor Ring Length

This field sets the maximum number of Tx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors. It is recommended to put a minimum ring descriptor length of 4.

For example, you can program any value up to 0x3FF in this field. This field is 10 bits wide, if you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **ARBS[7:0]:** Alternate Receive Buffer Size

Indicates size in bytes for Buffer 1 when ARBS[7:0] is programmed to a non-zero value.

When ARBS[7:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of Channel receive control register (ETH_DMACRXCR).

Bits 15:10  Reserved, must be kept at reset value.

Bits 9:0  **RDRL[9:0]:** Receive Descriptor Ring Length

This register sets the maximum number of Rx descriptors in the circular descriptor ring. The maximum number of descriptors is limited to 1K descriptors.

For example, you can program any value up to 0x3FF in this field. This field is 10-bit wide. If you program 0x3FF, you can have 1024 descriptors. If you want to have 10 descriptors, program it to a value of 0x9.

**Channel interrupt enable register (ETH_DMACIER)**

Address offset: 0x1134

Reset value: 0x0000 0000

The Channel Interrupt Enable register enables the interrupts reported by the Status register.
RM0477 Ethernet (ETH): media access control (MAC) with DMA controller

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 NIE: Normal Interrupt Summary Enable
When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the Channel status register (ETH_DMACSR):
- Bit 0: Transmit Interrupt
- Bit 2: Transmit Buffer Unavailable
- Bit 6: Receive Interrupt
- Bit 11: Early Receive Interrupt
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 AIE: Abnormal Interrupt Summary Enable
When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the Channel status register (ETH_DMACSR):
- Bit 1: Transmit Process Stopped
- Bit 7: Rx Buffer Unavailable
- Bit 8: Receive Process Stopped
- Bit 9: Receive Watchdog Timeout
- Bit 10: Early Transmit Interrupt
- Bit 12: Fatal Bus Error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 CDEE: Context Descriptor Error Enable
When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 FBEE: Fatal Bus Error Enable
When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 ERIE: Early Receive Interrupt Enable
When this bit is set along with the NIE bit, the Early Receive interrupt is enabled. When this bit is reset, the Early Receive interrupt is disabled.

Bit 10 ETIE: Early Transmit Interrupt Enable
When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 RWTE: Receive Watchdog Timeout Enable
When this bit is set along with the AIE bit, the Receive Watchdog Timeout interrupt is enabled. When this bit is reset, the Receive Watchdog Timeout interrupt is disabled.

Bit 8 RSE: Receive Stopped Enable
When this bit is set along with the AIE bit, the Receive Stopped interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 RBUE: Receive Buffer Unavailable Enable
When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.
Bit 6 **RIE**: Receive Interrupt Enable
When this bit is set along with the NIE bit, the Receive Interrupt is enabled. When this bit is reset, the Receive Interrupt is disabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable Enable
When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

Bit 1 **TXSE**: Transmit Stopped Enable
When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt Enable
When this bit is set along with the NIE bit, the Transmit Interrupt is enabled. When this bit is reset, the Transmit Interrupt is disabled.
Figure 988. Generation of ETH_DMAISR flags
Channel Rx interrupt watchdog timer register (ETH_DMACRXIWTR)

Address offset: 0x1138
Reset value: 0x0000 0000

The Receive Interrupt Watchdog Timer register indicates the watchdog timeout for Receive Interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the Channel status register (ETH_DMACSR).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RWTU[1:0]</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 17:16 **RWTU[1:0]**: Receive Interrupt Watchdog Timer Count Units

This field indicates the number of system clock cycles corresponding to one unit in RWT[7:0] field.

- 00: 256
- 01: 512
- 10: 1024
- 11: 2048

For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 × 512 = 1024 system clock cycles.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This field indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.

The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMACSR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].

When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.
Channel current application transmit descriptor register
(ETH_DMACCATXDR)

Address offset: 0x1144
Reset value: 0x0000 0000

The Channel Current Application Transmit Descriptor register points to the current Transmit descriptor read by the DMA.

Channel current application receive descriptor register
(ETH_DMACCARXDR)

Address offset: 0x114C
Reset value: 0x0000 0000

The Channel Current Application Receive Descriptor register points to the current Receive descriptor read by the DMA.

Bits 31:0 CURTDESAPTR[31:0]: Application Transmit Descriptor Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Bits 31:0 CURRDESAPTR[31:0]: Application Receive Descriptor Address Pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.
**Channel current application transmit buffer register (ETH_DMACCATXBR)**

Address offset: 0x1154  
Reset value: 0x0000 0000  
The Channel Current Application Transmit Buffer Address register points to the current Tx buffer address read by the DMA.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 CURTBUFAPTR[31:0]: Application Transmit Buffer Address Pointer  
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

**Channel current application receive buffer register (ETH_DMACCARXBR)**

Address offset: 0x115C  
Reset value: 0x0000 0000  
The Channel Current Application Receive Buffer Address register points to the current Rx buffer address read by the DMA.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 CURRBUFAPTR[31:0]: Application Receive Buffer Address Pointer  
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

**Channel status register (ETH_DMACSR)**

Address offset: 0x1160  
Reset value: 0x0000 0000  
The software driver (application) reads the Status register during interrupt service routine or polling to determine the status of the DMA.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>re</td>
<td>re</td>
<td>re</td>
<td>re</td>
<td>re</td>
<td>re</td>
<td>re</td>
<td>re</td>
<td>re</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NIS</th>
<th>AIS</th>
<th>CDE</th>
<th>FBE</th>
<th>ERI</th>
<th>ETI</th>
<th>RWT</th>
<th>RPS</th>
<th>RBU</th>
<th>RI</th>
<th>res</th>
<th>Res</th>
<th>res</th>
<th>TBU</th>
<th>TPS</th>
<th>TI</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rw</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
<td>rc_w1</td>
</tr>
</tbody>
</table>

3422/3791  
RM0477 Rev 6
Bits 31:22  Reserved, must be kept at reset value.

Bits 21:19  **REB[2:0]: Rx DMA Error Bits**
This field indicates the type of error that caused a bus error. For example, error response on the AHB interface.
Bit [2]: Error during data transfer by Rx DMA when 1, no error during data transfer by Rx DMA when 0.
Bit[1]: Error during descriptor access when 1, error during data buffer access when 0
Bit[0]: Error during read transfer when 1, error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16  **TEB[2:0]: Tx DMA Error Bits**
This field indicates the type of error that caused a bus error. For example, error response on the AHB interface.
Bit[2]: Error during data transfer by Tx DMA when 1, no error during data transfer by Tx DMA when 0
Bit[1]: Error during descriptor access when 1, error during data buffer access when 0
Bit[0]: Error during read transfer when 1, Error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15  **NIS**: Normal Interrupt Summary
Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
Bit 0: Transmit Interrupt
Bit 2: Transmit Buffer Unavailable
Bit 6: Receive Interrupt
Bit 11: Early Receive Interrupt
Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMACIER register) affect the Normal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14  **AIS**: Abnormal Interrupt Summary
Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACIER register:
Bit 1: Transmit Process Stopped
Bit 7: Receive Buffer Unavailable
Bit 8: Receive Process Stopped
Bit 10: Early Transmit Interrupt
Bit 12: Fatal Bus Error
Bit 13: Context Descriptor Error
Only unmasked bits affect the Abnormal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13  **CDE**: Context Descriptor Error
This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one's descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12  **FBE**: Fatal Bus Error
This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.
Bit 11 **ERI**: Early Receive Interrupt
   This bit indicates that the DMA filled the first data buffer of the packet. The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early Transmit Interrupt
   This bit indicates that the packet to be transmitted is fully transferred to the MTL Tx FIFO.

Bit 9 **RWT**: Receive Watchdog Timeout
   This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.

Bit 8 **RPS**: Receive Process Stopped
   This bit is asserted when the Rx process enters the Stopped state.

Bit 7 **RBU**: Receive Buffer Unavailable
   This bit indicates that the application owns the next descriptor in the Receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the Receive Descriptor Tail Pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.

Bit 6 **RI**: Receive Interrupt
   This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.
   The reception remains in the Running state.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit Buffer Unavailable
   This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the Debug status register (ETH_DMADSR) register explains the Transmit Process state transitions.
   To resume processing the Transmit descriptors, the application should do the following:
   1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
   2. Issue a Transmit Poll Demand command.
   For ring mode, the application should advance the Transmit Descriptor Tail Pointer register of a channel.

Bit 1 **TPS**: Transmit Process Stopped
   This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit Interrupt
   This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.
Channel missed frame count register (ETH_DMACMFCR)

Address offset: 0x116C
Reset value: 0x000 0000

This register has the number of packet counter that got dropped by the DMA either due to bus error or due to programing RPF field in Channel receive control register (ETH_DMACRXCR) register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r | rc_r |

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter
When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped Packet Counters
This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in Channel receive control register (ETH_DMACRXCR). The counter gets cleared when this register is read.
### Ethernet DMA register map and reset values

**Table 749. ETH_DMA common register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1000</td>
<td>ETH_DMAMR</td>
<td></td>
</tr>
<tr>
<td>0x1004</td>
<td>ETH_DMASBMR</td>
<td></td>
</tr>
<tr>
<td>0x1008</td>
<td>ETH_DMAISR</td>
<td></td>
</tr>
<tr>
<td>0x100C</td>
<td>ETH_DMADSR</td>
<td></td>
</tr>
</tbody>
</table>

**Table 750. ETH_DMA_CH register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x1100</td>
<td>ETH_DMACCR</td>
<td></td>
</tr>
<tr>
<td>0x1104</td>
<td>ETH_DMACTXCR</td>
<td></td>
</tr>
<tr>
<td>0x1108</td>
<td>ETH_DMACRXCR</td>
<td></td>
</tr>
<tr>
<td>0x1110</td>
<td>ETH_DMADXR</td>
<td></td>
</tr>
<tr>
<td>0x1114</td>
<td>ETH_DMACTXDLAR</td>
<td>TDESLA[31:0]</td>
</tr>
<tr>
<td>0x1118</td>
<td>ETH_DMACTXDTPR</td>
<td>TDT[31:0]</td>
</tr>
<tr>
<td>0x111C</td>
<td>ETH_DMACRXDLAR</td>
<td>RDESLA[31:0]</td>
</tr>
</tbody>
</table>

---
Table 750. ETH_DMA_CH register map and reset values (continued)

| Offset | Register name      | reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|-------------------|-------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x1124 | Reserved          |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1128 | ETH_DMACRXDTPR    | RDT[31:0]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x112C | ETH_DMACTXRLR     | TDRL[9:0]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1130 | ETH_DMACRXRLR     | ARBS[7:0]   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1134 | ETH_DMACIER       |             | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1138 | ETH_DMACRXIWTR    | RWT[15:10]  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x113C-0x1140 | Reserved         |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1144 | ETH_DMACCATXDR    | CURTDESAPTR[31:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1148 | Reserved          |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x114C | ETH_DMACCATXDR    | CURRDESAPTR[31:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1150 | Reserved          |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x1154 | ETH_DMACCATXBR    | CURTBUFAPTR[31:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1158 | Reserved          |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x115C | ETH_DMACCATXBR    | CURRBUFAPTR[31:0] | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1160 | ETH_DMACSR        | REB[2:0]    | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x1164-0x1168 | Reserved         |             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
Refer to Section 2.3 on page 149 for the register boundary addresses.
64.11.3 Ethernet MTL registers

Operating mode register (ETH_MTLOMR)

Address offset: 0x0C00
Reset value: 0x0000 0000

The Operating Mode register establishes the Transmit and Receive operating modes and commands.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 9 CNTCLR**: Counters Reset
  - When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.
  - If this bit is set along with CNTPRST bit, CNTPRST has precedence.

- **Bit 8 CNTPRST**: Counters Preset
  - When this bit is set:
    - **Tx queue underflow register (ETH_MTLTXQUR)** is initialized/preset to 0x7F0.
    - Missed Packet and Overflow Packet counters in **Rx queue missed packet and overflow counter register (ETH_MTLRXQMPQOCR)** is initialized/preset to 0x7F0
  - This bit is cleared automatically.

- **Bit 7 Reserved, must be kept at reset value.**

Bits 6:2  Reserved, must be kept at reset value.

- **Bit 1 DTXSTS**: Drop Transmit Status
  - When this bit is set, the Tx packet status received from the MAC is dropped in the MTL.
  - When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

- **Bit 0 Reserved, must be kept at reset value.**
Interrupt status register (ETH_MTLISR)

Address offset: 0x0C20
Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

<table>
<thead>
<tr>
<th>Bits 31:1</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td><strong>Q0IS</strong>: Queue interrupt status</td>
</tr>
<tr>
<td>This bit indicates that an interrupt has been generated by Queue. To reset this bit, read ETH_MTLQICSR register to identify the interrupt cause and clear the source.</td>
<td></td>
</tr>
</tbody>
</table>

Tx queue operating mode register (ETH_MTLTXQOMR)

Address offset: 0x0D00
Reset value: 0x0000 0008

The Queue Transmit Operating Mode register establishes the Transmit queue operating modes and commands.

<table>
<thead>
<tr>
<th>Bits 31:19</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 18:16</td>
<td><strong>TQS[2:0]</strong>: Transmit queue size</td>
</tr>
<tr>
<td>This field indicates the size of the allocated transmit queues in blocks of 256 bytes.</td>
<td></td>
</tr>
<tr>
<td>Queue size range from 256 bytes (TQS=0b000) to 2048 bytes (TQS=0b111).</td>
<td></td>
</tr>
<tr>
<td>Bits 15:7</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bits 6:4 TTC[2:0]: Transmit Threshold Control
These bits control the threshold level of the MTL Tx queue. The transmission starts when
the packet size within the MTL Tx queue is larger than the threshold. In addition, full
packets with length less than the threshold are also transmitted. These bits are used only
when the TSF bit is reset.
000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 TXQEN[1:0]: Transmit Queue Enable
This field is used to enable/disable the transmit queue.
00: Not enabled
10: Enabled
Others: Reserved, must not be used.
Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the
Tx queue by programming this field.

Bit 1 TSF: Transmit Store and Forward
When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue.
When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This
bit should be changed only when the transmission is stopped.

Bit 0 FTQ: Flush Transmit Queue
When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all
the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing
operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQOMR
register. The data which is already accepted by the MAC transmitter is not flushed. It is
scheduled for transmission and results in underflow and runt packet transmission.
Note: The flush operation is complete only when the Tx queue is empty and the application
has accepted the pending Tx Status of all transmitted packets. To complete this flush
operation, the PHY Tx clock (eth_tx_clk) should be active.

Tx queue underflow register (ETH_MTLTXQUR)
Address offset: 0x0D04
Reset value: 0x0000 0000
The Queue Underflow Counter register contains the counter for packets aborted because of
Transmit queue underflow and packets missed because of Receive queue packet flush

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
</tr>
<tr>
<td>rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f</td>
</tr>
</tbody>
</table>
The Queue Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

**Bits 31:12** Reserved, must be kept at reset value.

**Bit 11** **UFCNTOVF**: Overflow Bit for Underflow Packet Counter

This bit is set every time the Tx queue Underflow Packet Counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

**Bits 10:0** **UFFRMCNT[10:0]**: Underflow Packet Counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

**Tx queue debug register (ETH_MTLTXQDR)**

Address offset: 0x0D08

Reset value: 0x0000 0000

The Queue Transmit Debug register gives the debug status of various blocks related to the Transmit queue.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 22:20** **STXSTSF[2:0]**: Number of Status Words in Tx Status FIFO of Queue

This field indicates the current number of status in the Tx Status FIFO of this queue. When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

**Bit 19** Reserved, must be kept at reset value.

**Bits 18:16** **PTXQ[2:0]**: Number of Packets in the Transmit Queue

This field indicates the current number of packets in the Tx queue. When the DTXSTS bit of Operating mode register (ETH_MTLOMR) register is set to 1, this field does not reflect the number of packets in the Transmit queue.

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

**Bit 5** **TXSTSFSTS**: MTL Tx Status FIFO Full Status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

**Bit 4** **TXQSTS**: MTL Tx Queue Not Empty Status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.
Bit 3 **TWCSTS**: MTL Tx Queue Write Controller Status
When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx Queue Read Controller Status
This field indicates the state of the Tx Queue Read Controller:
00: Idle state
01: Read state (transferring data to the MAC transmitter)
10: Waiting for pending Tx Status from the MAC transmitter
11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit Queue in Pause
When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:
- Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
- Reception of 802.3x Pause packet when PFC is disabled

**Queue interrupt control status register (ETH_MTLQICSR)**
Address offset: 0x0D2C
Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue interrupts.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>15</strong></td>
<td><strong>14</strong></td>
<td><strong>13</strong></td>
<td><strong>12</strong></td>
<td><strong>11</strong></td>
<td><strong>10</strong></td>
<td><strong>9</strong></td>
<td><strong>8</strong></td>
<td><strong>7</strong></td>
<td><strong>6</strong></td>
<td><strong>5</strong></td>
<td><strong>4</strong></td>
<td><strong>3</strong></td>
<td><strong>2</strong></td>
<td><strong>1</strong></td>
<td><strong>0</strong></td>
</tr>
<tr>
<td><strong>RXUFIS</strong></td>
<td><strong>rc_w1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive Queue Overflow Interrupt Enable
When this bit is set, the Receive Queue Overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

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

Bit 16 **RXOVFIS**: Receive Queue Overflow Interrupt Status
This bit indicates that the Receive Queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:9 Reserved, must be kept at reset value.
### Bit 8  **TXUIE**: Transmit Queue Underflow Interrupt Enable

- When this bit is set, the Transmit Queue Underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

### Bits 7:1  Reserved, must be kept at reset value.

### Bit 0  **TXUNFIS**: Transmit Queue Underflow Interrupt Status

- This bit indicates that the Transmit Queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.
**Rx queue operating mode register (ETH_MTLRXQOMR)**

Address offset: 0x0D30  
Reset value: 0x0070 0000

The Queue Receive operating Mode register establishes the Receive queue operating modes and command.

| Bits 31:23 | Reserved, must be kept at reset value. |
| Bits 22:20 | **RQS[2:0]**: Receive Queue Size  
This field is read-only and the configured Rx FIFO size in blocks of 256 bytes is reflected in the reset value. The size of the Queue is (RQS + 1) × 256 bytes. |
| Bits 19:7 | Reserved, must be kept at reset value. |
| Bit 6 | **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets  
When this bit is set, the MAC does not drop the packets which only have the errors detected by the Receive Checksum Offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.  
When this bit is reset, all error packets are dropped if the FEP bit is reset. |
| Bit 5 | **RSF**: Receive Queue Store and Forward  
When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register. |
| Bit 4 | **FEP**: Forward Error Packets  
When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.  
When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA. |
Bit 3 **FUP**: Forward Undersized Good Packets

When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control

These bits control the threshold level of the MTL Rx queue (in bytes):

- 00: 64
- 01: 32
- 10: 96
- 11: 128

The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.

This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.
Rx queue missed packet and overflow counter register
(ETH_MTLRXQMPOCR)

Address offset: 0x0D34

Reset value: 0x0000 0000

The Queue missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

<table>
<thead>
<tr>
<th>Field</th>
<th>Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISCNTTOVF</td>
<td>0x0D34</td>
<td>Missed Packet Counter Overflow Bit</td>
</tr>
<tr>
<td>MISPKTCNT[10:0]</td>
<td></td>
<td>Missed Packet Counter</td>
</tr>
<tr>
<td>OVFCNTTOVF</td>
<td>0x0D34</td>
<td>Overflow Counter Overflow Bit</td>
</tr>
<tr>
<td>OVFPKTCNT[10:0]</td>
<td></td>
<td>Overflow Packet Counter</td>
</tr>
</tbody>
</table>

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTTOVF**: Missed Packet Counter Overflow Bit
When set, this bit indicates that the Rx Queue Missed Packet Counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed Packet Counter
This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.
This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

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

Bit 11 **OVFCNTTOVF**: Overflow Counter Overflow Bit
When set, this bit indicates that the Rx Queue Overflow Packet Counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow Packet Counter
This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.
**Rx queue debug register (ETH_MTLRXQDR)**

Address offset: 0x0D38  
Reset value: 0x0000 0000  
The Queue Receive Debug register gives the debug status of various blocks related to the Receive queue.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
<td>PRXQ[13:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

**Bits 31:30**  Reserved, must be kept at reset value.

**Bits 29:16**  **PRXQ[13:0]: Number of Packets in Receive Queue**  
This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256 Kbyte/16 bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

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

**Bits 5:4**  **RXQSTS[1:0]: MTL Rx Queue Fill-Level Status**  
This field gives the status of the fill-level of the Rx queue:  
00: Rx queue empty  
01: Rx queue fill-level below flow-control deactivate threshold  
10: Rx queue fill-level above flow-control activate threshold  
11: Rx queue full

**Bit 3**  Reserved, must be kept at reset value.

**Bits 2:1**  **RRCSTS[1:0]: MTL Rx Queue Read Controller State**  
This field gives the state of the Rx queue Read controller:  
00: Idle state  
01: Reading packet data  
10: Reading packet status (or timestamp)  
11: Flushing the packet data and status

**Bit 0**  **RWCSTS: MTL Rx Queue Write Controller Active Status**  
When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.
## Ethernet MTL register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0C00</td>
<td>ETH_MTLOMR</td>
<td>0x0C00</td>
<td>0x0C00</td>
<td>ETH_MTLOMR</td>
<td>0x0C00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0C00</td>
<td>ETH_MTLOMR</td>
<td>0x0C00</td>
</tr>
<tr>
<td>0x0C04-0x0C1C</td>
<td>Reserved</td>
<td>0x0000</td>
<td>0x0C04-0x0C1C</td>
<td>Reserved</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0C20</td>
<td>ETH_MTLISR</td>
<td>0x0C04</td>
<td>0x0C20</td>
<td>ETH_MTLISR</td>
<td>0x0C04</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0C20</td>
<td>ETH_MTLISR</td>
<td>0x0C04</td>
</tr>
<tr>
<td>0x0C24-0x0CFC</td>
<td>Reserved</td>
<td>0x0000</td>
<td>0x0C24-0x0CFC</td>
<td>Reserved</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0C40</td>
<td>Reserved</td>
<td>0x0000</td>
<td>0x0C40</td>
<td>Reserved</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0D00</td>
<td>ETH_MTLTXQOMR</td>
<td>0x0000</td>
<td>0x0D00</td>
<td>ETH_MTLTXQOMR</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0D00</td>
<td>ETH_MTLTXQOMR</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0D04</td>
<td>ETH_MTLTXQUR</td>
<td>0x0000</td>
<td>0x0D04</td>
<td>ETH_MTLTXQUR</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0D04</td>
<td>ETH_MTLTXQUR</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0D08</td>
<td>ETH_MTLTXQDR</td>
<td>0x0000</td>
<td>0x0D08</td>
<td>ETH_MTLTXQDR</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0D08</td>
<td>ETH_MTLTXQDR</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0D0C-0x0D28</td>
<td>Reserved</td>
<td>0x0000</td>
<td>0x0D0C-0x0D28</td>
<td>Reserved</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0D2C</td>
<td>ETH_MTLQICSR</td>
<td>0x0000</td>
<td>0x0D2C</td>
<td>ETH_MTLQICSR</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0D2C</td>
<td>ETH_MTLQICSR</td>
<td>0x0000</td>
</tr>
<tr>
<td>0x0D30</td>
<td>ETH_MTLRXQOMR</td>
<td>0x0000</td>
<td>0x0D30</td>
<td>ETH_MTLRXQOMR</td>
<td>0x0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0D30</td>
<td>ETH_MTLRXQOMR</td>
<td>0x0000</td>
</tr>
</tbody>
</table>
Refer to Section 2.3 on page 149 for the register boundary addresses.

| Offset | Register name reset value | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xD34  | ETH_MTLRXQPMPO CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | MISPKTCNT[10:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | OVFPKTCNT[10:0]          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xD38  | ETH_MTLRXQDR             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | PRXQ[13:0]               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value              | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xD3C- | Reserved                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 751. ETH_MTL register map and reset values (continued)
### 64.11.4 Ethernet MAC and MMC registers

**Operating mode configuration register (ETH_MACCR)**

Address offset: 0x0000  
Reset value: 0x0000 8000

The MAC Configuration register establishes the operating mode of the MAC.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>ARPEN: ARP Offload Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.</td>
</tr>
<tr>
<td></td>
<td>When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 30:28</th>
<th>SARC[2:0]: Source Address Insertion or Replacement Control</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:</td>
</tr>
<tr>
<td>010</td>
<td>the MAC inserts the content of the MAC Address 0 registers (MAC Address 0 high register (ETH_MACA0HR) and MAC Address x low register (ETH_MACxLR)) in the SA field of all transmitted packets.</td>
</tr>
<tr>
<td>011</td>
<td>the MAC replaces the content of the MAC Address 0 registers (MAC Address 0 high register (ETH_MACA0HR) and MAC Address x low register (ETH_MACxLR)) in the SA field of all transmitted packets.</td>
</tr>
<tr>
<td>110</td>
<td>the MAC inserts the content of the MAC Address 1 registers (MAC Address x high register (ETH_MACxHR) and MAC Address x low register (ETH_MACxLR)) in the SA field of all transmitted packets</td>
</tr>
<tr>
<td>111</td>
<td>the MAC replaces the content of the MAC Address 1 registers (MAC Address x high register (ETH_MACxHR) and MAC Address x low register (ETH_MACxLR)) in the SA field of all transmitted packets</td>
</tr>
<tr>
<td>Others</td>
<td>Reserved, must not be used.</td>
</tr>
</tbody>
</table>

**Note:** Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
Bit 27 **IPC**: Checksum Offload
When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.
The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap
These bits control the minimum IPG between packets during transmission.
000: 96 bit times
001: 88 bit times
010: 80 bit times
... 111: 40 bit times
This range of minimum IPG is valid in Full-duplex mode.
In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.
When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.
The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable
When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.
When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).
The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets
When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.
When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see Table 752: Giant Packet Status based on S2KP and JE Bits.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets
When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver.
This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 753: Packet Length based on the CST and ACS bits.
Bit 20 **ACS**: Automatic Pad or CRC Stripping  
When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.  
When this bit is reset, the MAC passes all incoming packets to the application, without any modification.  
*Note:* For information about how the settings of CST bit and this bit impact the packet length, see Table 753: Packet Length based on the CST and ACS bits.  

Bit 19 **WD**: Watchdog Disable  
When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.  
When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.  

Bit 18 Reserved, must be kept at reset value.  

Bit 17 **JD**: Jabber Disable  
When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.  
When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.  

Bit 16 **JE**: Jumbo Packet Enable  
When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.  
For more information about how the setting of this bit and the JE bit impact the Giant packet status, see Table 752: Giant Packet Status based on S2KP and JE Bits.  

Bit 15 Reserved, must be kept at reset value.  

Bit 14 **FES**: MAC Speed  
This bit selects the speed in the 10/100 Mbps mode:  
0: 10 Mbps  
1: 100 Mbps  

Bit 13 **DM**: Duplex Mode  
When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.  

Bit 12 **LM**: Loopback Mode  
When this bit is set, the MAC operates in the loopback mode at MII. The MII Rx clock input (eth_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.  

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode  
When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.  
When this bit is reset, the MAC transmitter ignores the status of the CRS signal.  

Bit 10 **DO**: Disable Receive Own  
When this bit is set, the MAC disabled the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.  
This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.
Bit 9  **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the MII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8  **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7  Reserved, must be kept at reset value.

Bits 6:5  **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (512 bit times for 10/100 Mbps) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:

- **00**: k = min (n, 10)
- **01**: k = min (n, 8)
- **10**: k = min (n, 4)
- **11**: k = min (n, 1)

where n = retransmission attempt

The random integer r takes the value in the range $0 \leq r < 2^k$.

This bit is applicable only in the Half-duplex mode.

Bit 4  **DC**: Deferral Check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbps mode.

Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on MII.

The deferral time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the Half-duplex mode.

Bits 3:2  **PRELEN[1:0]**: Preamble Length for Transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.

- **00**: 7 bytes of preamble
- **01**: 5 bytes of preamble
- **10**: 3 bytes of preamble
- **11**: Reserved, must not be used
Bit 1 TE: Transmitter Enable
When this bit is set, the Tx state machine of the MAC is enabled for transmission on the MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 RE: Receiver Enable
When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the MII interface.

Table 752 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 752. Giant Packet Status based on S2KP and JE Bits\(^{(1)}\)

<table>
<thead>
<tr>
<th>Length/Type Field</th>
<th>Received Packet Length</th>
<th>S2KP</th>
<th>JE</th>
<th>Giant Packet Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Untagged packet</td>
<td>&gt; 1,518</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 2,000</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 9,018</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>VLAN tagged packet</td>
<td>&gt; 1,522</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 2,000</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>&gt; 9,022</td>
<td>x</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

1. For all other combinations, the Giant Packet status is 0.

Table 753 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 753. Packet Length based on the CST and ACS bits

<table>
<thead>
<tr>
<th>Received Packet Length</th>
<th>CST</th>
<th>ACS</th>
<th>FCS Stripping Done</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 1,536</td>
<td>x</td>
<td>0</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>x</td>
<td>1</td>
<td>Yes (for Ethernet packets)</td>
</tr>
<tr>
<td>≥ 1,536</td>
<td>0</td>
<td>x</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>x</td>
<td>Yes (for Type packets)</td>
</tr>
</tbody>
</table>
Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004
Reset value: 0x0000 0000

The MAC Extended Configuration register establishes the operating mode of the MAC.

Bits 31:30  Reserved, must be kept at reset value.

Bits 29:25  **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in Operating mode configuration register (ETH_MACCR), gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:
- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24  **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in Operating mode configuration register (ETH_MACCR) together as minimum IPG greater than 96 bit times in steps of 8 bit times.

When this bit is reset, the MAC ignores EIPG field and interprets IPG field in Operating mode configuration register (ETH_MACCR) as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

**Note**: The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19  Reserved, must be kept at reset value.

Bit 18  **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the MAC Address 0 high register (ETH_MACA0HR) and MAC Address 0 low register MAC Address x low register (ETH_MACAxLR). The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).

When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.
Bit 17 **SPEN**: Slow Protocol Detection Enable
When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes. When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets
When this bit is set, the MAC receiver does not check the CRC field in the received packets. When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

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

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit
If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.
For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

### Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008
Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td><strong>RA</strong>: Receive All</td>
</tr>
<tr>
<td>30</td>
<td>receives all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word. When this bit is reset, the Receiver module passes only those packets to the application that pass the SA or DA address filter.</td>
</tr>
<tr>
<td>30:22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td><strong>DNTU</strong>: Drop Non-TCP/UDP over IP Packets</td>
</tr>
<tr>
<td>20</td>
<td>When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forwards only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.</td>
</tr>
</tbody>
</table>
Bit 20  **IPFE**: Layer 3 and Layer 4 Filter Enable
- When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.
- When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17  Reserved, must be kept at reset value.

Bit 16  **VTFE**: VLAN Tag Filter Enable
- When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11  Reserved, must be kept at reset value.

Bit 10  **HPF**: Hash or Perfect Filter
- When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.
- When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9  **SAF**: Source Address Filter Enable
- When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.
- When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

*Note:* According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8  **SAIF**: SA Inverse Filtering
- When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.
- When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6  **PCF[1:0]**: Pass Control Packets
- These bits control the forwarding of all control packets (including unicast and multicast Pause packets).
- 00: The MAC filters all control packets from reaching the application.
- 01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.
- 10: The MAC forwards all control packets to the application even if they fail the Address filter.
- 11: The MAC forwards the control packets that pass the Address filter.

Bit 5  **DBF**: Disable Broadcast Packets
- When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.
- When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4  **PM**: Pass All Multicast
- When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is ‘1’) are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.
Bit 3 **DAIF**: DA Inverse Filtering
When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast
When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.
When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast
When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.
When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode
When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

**Watchdog timeout register (ETH_MACWTR)**
Address offset: 0x000C
Reset value: 0x0000 0000
The Watchdog Timeout register controls the watchdog timeout for received packets.
Bits 31:9  Reserved, must be kept at reset value.

Bit 8  PWE: Programmable Watchdog Enable
When this bit is set and the WD bit of the Operating mode configuration register (ETH_MACCR) register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in Operating mode configuration register (ETH_MACCR) register.

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:0  WTO[3:0]: Watchdog Timeout
When the PWE bit is set and the WD bit of the Operating mode configuration register (ETH_MACCR) register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.
Encoding is as follows:
- 0x0: 2 Kbytes
- 0x1: 3 Kbytes
- 0x2: 4 Kbytes
- 0x3: 5 Kbytes
- ...
- 0xC: 14 Kbytes
- 0xD: 15 Kbytes
- 0xE: 16383 Bytes
- 0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)
Address offset: 0x0010
Reset value: 0x0000 0000
The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).
The Hash table is used for group address filtering.
For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b10000 selects Bit 0 of the Hash Table Register 1.
The Hash value of the destination address is calculated in the following way:
1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.
If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.
Ethernet (ETH): media access control (MAC) with DMA controller

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014
Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determine the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 6'b10000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:
1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

Bits 31:0  **HT31T0[31:0]**: MAC Hash Table First 32 Bits
This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 2 register (ETH_MACHT2R)

Bits 31:0  **HT63T32[31:0]**: MAC Hash Table Second 32 Bits
This field contains the second 32 Bits [63:32] of the Hash table.
VLAN tag register (ETH_MACVTR)

Address offset: 0x0050
Reset value: 0x0000 0000

The VLAN Tag register identifies the IEEE 802.1Q VLAN type packets.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>EIVLRXS</td>
<td>Enable Inner VLAN Tag in Rx Status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, the MAC provides the inner VLAN Tag in the Rx status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29-28</td>
<td>EIVLS[1:0]</td>
<td>Enable Inner VLAN Tag Stripping on Receive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This field indicates the stripping operation on inner VLAN Tag in received packet:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00: Do not strip</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01: Strip if VLAN filter passes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10: Strip if VLAN filter fails</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11: Always strip</td>
</tr>
<tr>
<td>27</td>
<td>ERIVLT</td>
<td>Enable Inner VLAN Tag</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag if present. When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag if present. The ERSVLM bit determines which VLAN type is enabled for filtering or matching. The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.</td>
</tr>
<tr>
<td>26</td>
<td>EDVLP</td>
<td>Enable Double VLAN Processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).</td>
</tr>
<tr>
<td>25</td>
<td>VTHM</td>
<td>VLAN Tag Hash Table Match Enable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHT register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table. When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison. When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison. When this bit is reset, the VLAN Hash Match operation is not performed.</td>
</tr>
<tr>
<td>24</td>
<td>EIVLRXS</td>
<td>Enable VLAN Tag in Rx status</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is set, MAC provides the outer VLAN Tag in the Rx status.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bits 22:21  **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive

This field indicates the stripping operation on the outer VLAN Tag in received packet:
- 00: Do not strip
- 01: Strip if VLAN filter passes
- 10: Strip if VLAN filter fails
- 11: Always strip

Bit 20  **DOVLT**: Disable VLAN Type Check

When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19  **ERSVLM**: Enable Receive S-VLAN Match

When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18  **ESVL**: Enable S-VLAN

When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17  **VTIM**: VLAN Tag Inverse Match Enable

When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16  **ETV**: Enable 12-Bit VLAN Tag Comparison

When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:0  **VL[15:0]**: VLAN Tag Identifier for Receive Packets

This field contains the 802.1Q VLAN tag to identify the VLAN packets. This VLAN tag identifier is compared to the 15th and 16th bytes of the packets being received for VLAN packets. The following list describes the bits of this field:
- Bits[15:13]: User Priority
- Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
- Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

When the ETV bit is set, only the VID is used for comparison.
If this field ([11:0] if ETV is set) is all zeros, the MAC does not check the 15th and 16th bytes for VLAN tag comparison and declares all packets with Type field value of 0x8100 or 0x88a8 as VLAN packets.
VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058
Reset value: 0x0000 0000

When the VTHM bit of VLAN tag register (ETH_MACVTR) register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of VLAN tag register (ETH_MACVTR) register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

VLHT[15:0]

- Bits 31:16 Reserved, must be kept at reset value.
- Bits 15:0 VLHT[15:0]: VLAN Hash Table
  This field contains the 16-bit VLAN Hash Table.
VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060
Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the Transmit packets. It also contains the VLAN tag insertion controls.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
<th>rw</th>
</tr>
</thead>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLI**: VLAN Tag Input
When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSV**: C-VLAN or S-VLAN
When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN
1: S-LAN

Bit 18 **VLP**: VLAN Priority Control
When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets
00: No VLAN tag deletion, insertion, or replacement
01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.
10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.
11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

*Note*: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets
This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:

Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag
**Inner VLAN inclusion register (ETH_MACIVIR)**

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the Transmit packet. It also contains the inner VLAN tag insertion controls.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLT[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN Tag Input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN Priority Control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN Tag Control in Transmit Packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

**Note:** Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.
Bits 15:0 **VLT[15:0]**: VLAN Tag for Transmit Packets
This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:
- Bits[15:13]: User Priority
- Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
- Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

**Tx Queue flow control register (ETH_MACQTXFCR)**
Address offset: 0x0070
Reset value: 0x0000 0000
The Flow Control register controls the generation and reception of the Control (Pause Command) packets by the Flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PT[15:0]**: Pause Time
This field holds the value to be used in the Pause Time field in the Tx control packet.

Bits 15:8 Reserved, must be kept at reset value.

**Bit 7 DZPQ: Disable Zero-Quanta Pause**
When this bit is set, it disables the automatic generation of the zero-quanta Pause packets. When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.
Bits 6:4 \textbf{PLT}[2:0]: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)
001: Pause Time minus 28 Slot Times (PT -28 slot times)
010: Pause Time minus 36 Slot Times (PT -36 slot times)
011: Pause Time minus 144 Slot Times (PT -144 slot times)
100: Pause Time minus 256 Slot Times (PT -256 slot times)
101: Pause Time minus 512 Slot Times (PT -512 slot times)
110 to 111: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the MII interface. This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 \textbf{TFE}: Transmit Flow Control Enable

\textbf{Full-duplex mode:} when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

\textbf{Half-duplex mode:} when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 \textbf{FCB\_BPA}: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

\textbf{Full-Duplex mode:} this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

\textbf{Half-duplex mode:} When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.
**Rx flow control register (ETH_MACRXFCR)**

Address offset: 0x0090  
Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC Transmit based on the received Pause packet.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 1  UP: Unicast Pause Packet Detect**

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in *MAC Address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)* and *MAC Address x low register (ETH_MACAxLR)*.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

**Note:** The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01 80 C2 00 00 01) is as specified in IEEE 802.1 Qbb-2011.

**Bit 0  RFE: Receive Flow Control Enable**

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the Transmit queue, with matching priorities, for a duration of received Pause time.
Interrupt status register (ETH_MACISR)

Address offset: 0x00B0
Reset value: 0x0000 0000

The Interrupt Status register contains the status of interrupts.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bit 14  **RXSTSIS:** Receive Status Interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the *Rx Tx status register (ETH_MACRXTXSR)*. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 13  **TXSTSIS:** Transmit Status Interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the *Rx Tx status register (ETH_MACRXTXSR)*:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber Timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.
Bit 12 **TSIS**: Timestamp Interrupt Status  
If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

- The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.
- There is an overflow in the Seconds register.
- The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)** and **Tx timestamp status seconds register (ETH_MACTXTSSSR)** registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)** and **Tx timestamp status seconds register (ETH_MACTXTSSSR)** registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of **CSR software control register (ETH_MACCSRSWCR)** is set) in the **Timestamp status register (ETH_MACTSSR)**.

Bit 11 **Reserved**, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC Transmit Interrupt Status  
This bit is set high when an interrupt is generated in the **MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)**. This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive Interrupt Status  
This bit is set high when an interrupt is generated in the **MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)**. This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMS**: MMC Interrupt Status  
This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 **Reserved**, must be kept at reset value.

Bit 5 **LPIIS**: LPI Interrupt Status  
This bit is set for any LPI state entry or exit in the MAC Transmitter or Receiver. This bit is cleared when the TLPIEN bit of **LPI control and status register (ETH_MACLCSR)** is read.

Bit 4 **PMTIS**: PMT Interrupt Status  
This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the **PMT control status register (ETH_MACPCSR)**.

Bit 3 **PHYIS**: PHY Interrupt  
This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:0 **Reserved**, must be kept at reset value.
Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4
Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

<table>
<thead>
<tr>
<th>Bits</th>
<th>RXSTSIE</th>
<th>TXSTSIE</th>
<th>TSIE</th>
<th>LPIIE</th>
<th>PMTIE</th>
<th>PHYIE</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:15</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bit 14  RXSTSIE: Receive Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the Interrupt status register (ETH_MACISR).

Bit 13  TXSTSIE: Transmit Status Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the Interrupt status register (ETH_MACISR).

Bit 12  TSIE: Timestamp Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in Interrupt status register (ETH_MACISR).

Bits 11:6  Reserved, must be kept at reset value.

Bit 5  LPIIE: LPI Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in Interrupt status register (ETH_MACISR).

Bit 4  PMTIE: PMT Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in Interrupt status register (ETH_MACISR).

Bit 3  PHYIE: PHY Interrupt Enable

When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in Interrupt status register (ETH_MACISR).

Bits 2:0  Reserved, must be kept at reset value.
Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8
Reset value: 0x0000 0000

The Receive Transmit Status register contains the Receive and Transmit Error status.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:9</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>RWT: Receive Watchdog Timeout</td>
<td></td>
</tr>
<tr>
<td>7:6</td>
<td>Reserved, must be kept at reset value.</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>EXCOL: Excessive Collisions</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>LCOL: Late Collision</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>EXDEF: Excessive Deferral</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 RWT: Receive Watchdog Timeout

This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the Operating mode configuration register (ETH_MACCR). This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the Operating mode configuration register (ETH_MACCR).

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 EXCOL: Excessive Collisions

When the DTXSTS bit is set in the Operating mode register (ETH_MTLOMR), this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the Operating mode configuration register (ETH_MACCR), this bit is set after the first collision and the packet transmission is aborted.

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bit 4 LCOL: Late Collision

When the DTXSTS bit is set in the Operating mode register (ETH_MTLOMR), this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode.

This bit is not valid if the Underflow error occurs.

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).

Bit 3 EXDEF: Excessive Deferral

When the DTXSTS bit is set in the Operating mode register (ETH_MTLOMR) and the DC bit is set in the Operating mode configuration register (ETH_MACCR), this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 when Jumbo packet is enabled).

Cleared on read (or write of 1 when RCWE bit in CSR software control register (ETH_MACCSRSWCR) is set).
Bit 2  **LCARR**: Loss of Carrier
When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.
 Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1  **NCARR**: No Carrier
When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.
 Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0  **TJT**: Transmit Jabber Timeout
This bit indicates that the Transmit Jabber Timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the *Operating mode configuration register (ETH_MACCR)*.
 Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

**PMT control status register (ETH_MACPCSR)**
Address offset: 0x00C0
Reset value: 0x0000 0000

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

Bit 31  **RWKFILTRST**: Remote Wake-up Packet Filter Register Pointer Reset
When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24  **RWKPTR[4:0]**: Remote wake-up FIFO Pointer
This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.
Bit 10  **RWKPFE**: Remote wake-up Packet Forwarding Enable
When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.

The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low.

The setting of this bit has no effect when PWRDWN is set high.

*Note: If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.*

Bit 9  **GLBLUCAST**: Global Unicast
When this bit is set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7  Reserved, must be kept at reset value.

Bit 6  **RWKPRCDV**: Remote wake-up Packet Received
When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5  **MGKPRCDV**: Magic Packet Received
When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in **CSR software control register (ETH_MACCSRWSWC)** is set).

Bits 4:3  Reserved, must be kept at reset value.

Bit 2  **RWKPKTEN**: Remote wake-up Packet Enable
When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1  **MGKPKTEN**: Magic Packet Enable
When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0  **PWRDWN**: Power Down
When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

*Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.*
Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>MACRWKPFR[31:16]</th>
<th>MACRWKPFR[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>1 5 1 4 1 3 1 2 1 1 1 0 9876543210</td>
<td>1 5 1 4 1 3 1 2 1 1 1 0 9876543210</td>
</tr>
</tbody>
</table>

Bits 31:0 MACRWKPFR[31:0]: Remote wake-up packet filter

Refer to Table 721, Table 722 and Table 723 for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.
Bits 31:22  Reserved, must be kept at reset value.

Bit 21  **LPITCSE**: LPI Tx Clock Stop Enable
When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20  **LPITE**: LPI Timer Enable
This bit controls the automatic entry of the MAC Transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC Transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.
After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the Transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.
When LPITE is 0, the LPI Auto timer is disabled and MAC Transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19  **LPITXA**: LPI Tx Automate
This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the Transmit side.
If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxQOMR, when the MAC is in the LPI mode, it exits the LPI mode.
When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18  Reserved, must be kept at reset value.

Bit 17  **PLS**: PHY Link Status
This bit indicates the link status of the PHY. The MAC Transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.
When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16  **LPIEN**: LPI Enable
When this bit is set, it instructs the MAC Transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.
This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10  Reserved, must be kept at reset value.

Bit 9  **RLPIST**: Receive LPI State
When this bit is set, it indicates that the MAC is receiving the LPI pattern on the MII interface.

Bit 8  **TLPIST**: Transmit LPI State
When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the MII interface.

Bits 7:4  Reserved, must be kept at reset value.
Bit 3  **RLPIEX**: Receive LPI Exit
When this bit is set, it indicates that the MAC Receiver has stopped receiving the LPI pattern on the MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

*Note:* This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2  **RLPIEN**: Receive LPI Entry
When this bit is set, it indicates that the MAC Receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

*Note:* This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1  **TLPIEX**: Transmit LPI Exit
When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

Bit 0  **TLPIEN**: Transmit LPI Entry
When this bit is set, it indicates that the MAC Transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

**LPI timers control register (ETH_MACLTCR)**
Address offset: 0x00D4
Reset value: 0x03E8 0000

The LPI Timers Control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.
LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8
Reset value: 0x0000 0000

This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of LPI control and status register (ETH_MACLCSR) register is set to 1.

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 LPIET[19:0]: LPI Entry Timer

This field specifies the time in microseconds the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

One-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC
Reset value: 0x0000 0063

This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.
Version register (ETH_MACVR)
Address offset: 0x0110
Reset value: 0x0000 3242

The version register identifies the version of the Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

USERVER[7:0]   SNPSVER[7:0]
rrrrrrrrrrrrrrrr

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:8  USERVER[7:0]: ST-defined version

Bits 7:0  SNPSVER[7:0]: IP version

Debug register (ETH_MACDR)
Address offset: 0x0114
Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

TPESTS

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

RPESTS

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  TIC_1US_CNTR[11:0]: 1 µs tick Counter
The application must program this counter so that the number of clock cycles of CSR clock is
1 µs (subtract 1 from the value before programming).
For example if the CSR clock is 100 MHz then this field needs to be programmed to
100 - 1 = 99 (which is 0x63).
This is required to generate the 1 µs events that are used to update some of the EEE related
counters.
Bits 31:19  Reserved, must be kept at reset value.

Bits 18:17  **TFCSTS[1:0]:** MAC Transmit Packet Controller Status
- This field indicates the state of the MAC Transmit Packet Controller module:
  - 00: Idle state
  - 01: Waiting for one of the following:
    - Status of the previous packet
    - IPG or backoff period to be over
  - 10: Generating and transmitting a Pause control packet (in Full-duplex mode)
  - 11: Transferring input packet for transmission

Bit 16  **TPESTS:** MAC MII Transmit Protocol Engine Status
- When this bit is set, it indicates that the MAC MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3  Reserved, must be kept at reset value.

Bits 2:1  **RFCFCSTS[1:0]:** MAC Receive Packet Controller FIFO Status
- When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC Receive Packet Controller module.

Bit 0  **RPESTS:** MAC MII Receive Protocol Engine Status
- When this bit is set, it indicates that the MAC MII receive protocol engine is actively receiving data, and it is not in the Idle state.

**HW feature 0 register (ETH_MACHWF0R)**

Address offset: 0x011C

Reset value: 0x0A0D 73F5

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.
Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **ACTPHYSEL[2:0]:** Active PHY Selected
   When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
   000: GMII or MII
   001: RGMII
   010: SGMII
   011: TBI
   100: RMII
   101: RTBI
   110: SMII
   Others: Reserved, must not be used

Bit 27  **SAVLAININS:** Source Address or VLAN Insertion Enable
   This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25  **TSSTSSEL[1:0]:** Timestamp System Time Source
   This bit indicates the source of the Timestamp system time:
   01: Internal
   10: External
   11: Both
   00: Reserved, must not be used
   This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24  **MACADR64SEL:** MAC Addresses 64-127 Selected
   This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23  **MACADR32SEL:** MAC Addresses 32-63 Selected
   This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18  **ADDMACADRSEL[4:0]:** MAC Addresses 1-31 Selected
   This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17  Reserved, must be kept at reset value.

Bit 16  **RXCOESEL:** Receive Checksum Offload Enabled
   This bit is set to 1 when the Enable Receive TCP/IP Checksum Check option is selected

Bit 15  Reserved, must be kept at reset value.

Bit 14  **TXCOESEL:** Transmit Checksum Offload Enabled
   This bit is set to 1 when the Enable Transmit TCP/IP Checksum Insertion option is selected

Bit 13  **EEESEL:** Energy Efficient Ethernet Enabled
   This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12  **TSSEL:** IEEE 1588-2008 Timestamp Enabled
   This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10  Reserved, must be kept at reset value.

Bit 9  **ARPOFFSEL:** ARP Offload Enabled
   This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8  **MMCSEL:** RMON Module Enable
   This bit is set to 1 when the Enable MAC management counters (MMC) option is selected
Bit 7 **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6 **RWKSEL**: PMT Remote Wake-up Packet Enable
This bit is set to 1 when the Enable Remote Wake-up Packet Detection option is selected

Bit 5 **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4 **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3 **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2 **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1 **GMIISEL**: 1000 Mbps Support
This bit is set to 1 when 1000 Mbps is selected as operating mode.

Bit 0 **MIISEL**: 10 or 100 Mbps Support
This bit is set to 1 when 10/100 Mbps is selected as operating mode.

**HW feature 1 register (ETH_MACHWF1R)**

Address offset: 0x0120

Reset value: 0x1104 1904

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.
Bit 31  Reserved, must be kept at reset value.

Bits 30:27  **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
   This field indicates the total number of L3 or L4 filters:
   0000: No L3 or L4 Filter
   0001: 1 L3 or L4 Filter
   0010: 2 L3 or L4 Filters
   ...
   1000: 8 L3 or L4

Bit 26  Reserved, must be kept at reset value.

Bits 25:24  **HASHTBLSZ[1:0]**: Hash Table Size
   This field indicates the size of the Hash table:
   00: No Hash table
   01: 64
   10: 128
   11: 256

Bit 23  **POUOST**: One Step for PTP over UDP/IP Feature Enable
   This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22  Reserved, must be kept at reset value.

Bit 21  **RAVSEL**: Rx Side Only AV Feature Enable
   This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20  **AVSEL**: AV Feature Enable
   This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19  **DBGMEMA**: DMA Debug Registers Enable
   This bit is set to 1 when the Debug Mode Enable option is selected.

Bit 18  **TSOEN**: TCP Segmentation Offload Enable
   This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected.

Bit 17  **SPHEN**: Split Header Feature Enable
   This bit is set to 1 when the Enable Split Header Structure option is selected.

Bit 16  **DCBEN**: DCB Feature Enable
   This bit is set to 1 when the Enable Data Center Bridging option is selected.

Bits 15:14  **ADDR64[1:0]**: Address width
   This field indicates the configured address width.
   00: 32 bits
   Others: Reserved, must not be used

Bit 13  **ADVTHWORD**: IEEE 1588 High Word Register Enable
   This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected.

Bit 12  **PTOEN**: PTP Offload Enable
   This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11  **OSTEN**: One-Step Timestamping Enable
   This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.
Bits 10:6 **TXFIFOSIZE[4:0]**: MTL Transmit FIFO Size

This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:

- 00000: 128 bytes
- 00001: 256 bytes
- 00010: 512 bytes
- 00011: 1,024 bytes
- 00100: 2,048 bytes
- 00101: 4,096 bytes
- 00110: 8,192 bytes
- 00111: 16,384 bytes
- 01000: 32 Kbytes
- 01001: 64 Kbytes
- 01010: 128 Kbytes
- 01011 to 11111: Reserved, must not be used

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL Receive FIFO Size

This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:

- 00000: 128 bytes
- 00001: 256 bytes
- 00010: 512 bytes
- 00011: 1,024 bytes
- 00100: 2,048 bytes
- 00101: 4,096 bytes
- 00110: 8,192 bytes
- 00111: 16,384 bytes
- 01000: 32 Kbytes
- 01001: 64 Kbytes
- 01010: 128 Kbytes
- 01011: 256 Kbytes
- 01100 to 11111: Reserved, must not be used
HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124
Reset value: 0x4100 0000

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

<table>
<thead>
<tr>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r r r r r r r r r r r r r r</td>
</tr>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
</tr>
<tr>
<td>r r r r r r r r r r r r r r r r r r r r r r</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:28  **AUXSNAPNUM[2:0]**: Number of Auxiliary Snapshot Inputs

This field indicates the number of auxiliary snapshot inputs:
- 000: No auxiliary input
- 001: 1 auxiliary input
- 010: 2 auxiliary inputs
- 011: 3 auxiliary inputs
- 100: 4 auxiliary inputs
- 101 to 111: Reserved, must not be used

Bit 27  Reserved, must be kept at reset value.

Bits 26:24  **PPSOUTNUM[2:0]**: Number of PPS Outputs

This field indicates the number of PPS outputs:
- 000: No PPS output
- 001: 1 PPS output
- 010: 2 PPS outputs
- 011: 3 PPS outputs
- 100: 4 PPS outputs
- 101 to 111: Reserved, must not be used

Bits 23:22  **TDCSZ[1:0]**: Tx DMA Descriptor Cache Size in terms of 16-byte descriptors

- 00: Cache not configured
- 01: Four 16-byte descriptors
- 10: Eight 16-byte descriptors
- 11: Sixteen 16-byte descriptors

Bits 21:18  **TXCHCNT[3:0]**: Number of DMA Transmit Channels

This field indicates the number of DMA Transmit channels:
- 0000: 1 DMA Tx Channel
- 0001: 2 DMA Tx Channels
- ...
- 0111: 8 DMA Tx
Bits 17:16  **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
   00: Cache not configured
   01: Four 16-byte descriptors
   10: Eight 16-byte descriptors
   11: Sixteen 16-byte descriptors

Bits 15:12  **RXCHCNT[3:0]**: Number of DMA Receive Channels
   This field indicates the number of DMA Receive channels:
   0000: 1 DMA Rx Channel
   0001: 2 DMA Rx Channels
   ..
   0111: 8 DMA Rx

Bits 11:10  Reserved, must be kept at reset value.

Bits 9:6  **TXQCNT[3:0]**: Number of MTL Transmit Queues
   This field indicates the number of MTL Transmit queues:
   0000: 1 MTL Tx queue
   0001: 2 MTL Tx queues
   ..
   0111: 8 MTL Tx

Bits 5:4  Reserved, must be kept at reset value.

Bits 3:0  **RXQCNT[3:0]**: Number of MTL Receive Queues
   This field indicates the number of MTL Receive queues:
   0000: 1 MTL Rx queue
   0001: 2 MTL Rx queues
   ..
   0111: 8 MTL Rx
HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128
Reset value: 0x0000 0020

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>31:6 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 5 **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4 **CBTISEL**: Queue/Channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx feature is selected.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **NRF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
- 000: No Extended Rx VLAN Filters
- 001: 4 Extended Rx VLAN Filters
- 010: 8 Extended Rx VLAN Filters
- 011: 16 Extended Rx VLAN Filters
- 100: 24 Extended Rx VLAN Filters
- 101: 32 Extended Rx VLAN Filters
- 110 to 111: Reserved, must not be used
**MDIO address register (ETH_MACMDIOAR)**

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO Address register controls the management cycles to external PHY through a management interface.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>PSE</th>
<th>BTB</th>
<th>PA[4:0]</th>
<th>RDA[4:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 31:28** Reserved, must be kept at reset value.

**Bit 27 PSE**: Preamble Suppression Enable

When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

**Bit 26 BTB**: Back to Back transactions

When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (MII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

**Bits 25:21 PA[4:0]**: Physical Layer Address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.

This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

**Bits 20:16 RDA[4:0]**: Register/Device Address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

**Bit 15** Reserved, must be kept at reset value.

**Bits 14:12 NTC[2:0]**: Number of Training Clocks

This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.
Bits 11:8 **CR[3:0]:** CSR Clock Range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency (eth_hclk):
- 0000: MDC clock = eth_hclk / 42
- 0001: MDC clock = eth_hclk / 62
- 0010: MDC clock = eth_hclk / 16
- 0011: MDC clock = eth_hclk / 26
- 0100: MDC clock = eth_hclk / 102
- 0101: MDC clock = eth_hclk / 124
- 0110 to 0111: Reserved, must not be used

The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range. When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits to 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3.

Program the following values only if the interfacing chips support faster MDC clocks:
- 1000: eth_hclk / 4
- 1001: eth_hclk / 6
- 1010: eth_hclk / 8
- 1011: eth_hclk / 10
- 1100: eth_hclk / 12
- 1101: eth_hclk / 14
- 1110: eth_hclk / 16
- 1111: eth_hclk / 18

Bits 7:5 **Reserved, must be kept at reset value.**

Bit 4 **SKAP:** Skip Address Packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]:** MII Operation Command

This bit indicates the operation command to the PHY.
- 00: Reserved, must not be used
- 01: Write
- 10: Post Read Increment Address for Clause 45 PHY
- 11: Read

When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E:** Clause 45 PHY Enable

When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.
Bit 0  **MB**: MII Busy

The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in **MDIO address register (ETH_MACMDIOAR)** and **MDIO data register (ETH_MACMDIODR)** as long as this bit is set.

For write transfers, the application must first write 16-bit data in the MD field (and also RA field when C45E is set) in **MDIO data register (ETH_MACMDIODR)** register before setting this bit. When C45E is set, it should also write into the RA field of **MDIO data register (ETH_MACMDIODR)** before initiating a read transfer. When a read transfer is completed (MII busy=0), the data read from the PHY register is valid in the MD field of the **MDIO data register (ETH_MACMDIODR)**.

Note: *Even if the addressed PHY is not present, there is no change in the functionality of this bit.*

**MDIO data register (ETH_MACMDIODR)**

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO Data register stores the Write data to be written to the PHY register located at the address specified in **MDIO address register (ETH_MACMDIOAR)**. This register also stores the Read data from the PHY register located at the address specified by MDIO Address register.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>MD[15:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Bits 31:16 RA[15:0]: Register Address**

This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

**Bits 15:0 MD[15:0]: MII Data**

This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.
ARP address register (ETH_MACARPAR)

Address offset: 0x0210
Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 ARPPA[31:0]: ARP Protocol Address
This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSCWR)

Address offset: 0x0230
Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bits clearing.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 SEEN: Slave Error Response Enable
When this bit is set, the MAC responds with a Slave Error for accesses to reserved registers in CSR space.
When this bit is reset, the MAC responds with an Okay response to any register accessed from CSR space.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 RCWE: Register Clear on Write 1 Enable
When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.
When this bit is reset, the access mode to these register fields remains rc_r (clear on read).
MAC Address 0 high register (ETH_MACA0HR)

Address offset: 0x0300
Reset value: 0x8000 FFFF

The MAC Address0 High register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the MII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x11 22 33 44 55 66 is received (0x11 in lane 0 of the first column) on the MII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x66 55 44 33 22 11.

If the MAC address registers are configured to be double-synchronized to the MII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

`ADDRHI[15:0]`

Bit 31  AE: Address Enable
This bit is always set to 1.

Bits 30:16  Reserved, must be kept at reset value.

Bits 15:0  ADDRHI[15:0]: MAC Address0[47:32]
This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

MAC Address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)
Reset value: 0xFFFF FFFF

The MAC Address x Low register holds the lower 32 bits of the 6-byte first MAC address of the station.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

`ADDRLO[31:16]`

`ADDRLO[15:0]`
Bits 31:0  **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)
This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the Transmit Flow Control (Pause) Packets.

**MAC Address x high register (ETH_MACAxHR)**
Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)
Reset value: 0x0000 FFFF
The MAC Address x High register holds the upper 16 bits of the second 6-byte MAC address of the station.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>AE</td>
<td>SA</td>
<td>MBC[5:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16
AE SA MBC[5:0]

Bit 31  **AE**: Address Enable
When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30  **SA**: Source Address
When this bit is set, the MAC Address x[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.
0: DA
1: SA

Bits 29:24  **MBC[5:0]**: Mask Byte Control
These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:
Bit 29: ETH_MACAxHR[15:8]
Bit 28: ETH_MACAxHR[7:0]
Bit 27: ETH_MACAxLR[31:24]
Bit 26: ETH_MACAxLR[23:16]
Bit 25: ETH_MACAxLR[15:8]
Bit 24: ETH_MACAxLR[7:0]
You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:16  Reserved, must be kept at reset value.

Bits 15:0  **ADDRHI[15:0]**: MAC Address1 [47:32]
This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.
MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700
Reset value: 0x0000 0000

This register configures the MMC operating mode.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets
The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.
When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of Packet filtering control register (ETH_MACPFR).
When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset
When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF FFF0 (Half 16).
When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF FFF0 (Full 16).
For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset
When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.
This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze
When this bit is set, it freezes all MMC counters to their current value.
Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.
Ethernet (ETH): media access control (MAC) with DMA controller RM0477

Bit 2 **RSTONRD**: Reset on Read
- When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset).
- The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover
- When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset
- When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

**MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)**

Address offset: 0x0704
Reset value: 0x0000 0000

This register maintains the interrupts generated from all Receive statistics counters.

The MMC Receive Interrupt register maintains the interrupts that are generated when the following occur:
- Receive statistic counters reach half of their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the CNTSTOPRO is set in **MMC control register (ETH_MMC_CONTROL)**, interrupts are set but the counter remains at all-ones. The MMC Receive Interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Reset Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>RXLPITRCIS: MMC Receive LPI transition counter interrupt status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>RXCGERPIS: MMC Receive CRC error counter interrupt status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:28  Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC Receive LPI transition counter interrupt status
- This bit is set when the **Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)** counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC Receive LPI microsecond counter interrupt status
- This bit is set when the **Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)** counter reaches half of the maximum value or the maximum value.

Bits 25:18  Reserved, must be kept at reset value.
Bit 17 **RXUCGPIS**: MMC Receive Unicast Good Packet Counter Interrupt Status
This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC Receive Alignment Error Packet Counter Interrupt Status
This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC Receive CRC Error Packet Counter Interrupt Status
This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

**MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)**

Address offset: 0x0708
Reset value: 0x0000 0000

This register maintains the interrupts generated from all Transmit statistics counters.

The MMC Transmit Interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF FFFF for 16-bit counter).

When **CNTSTOPRO** is set in **MMC control register (ETH_MMC_CONTROL)**, the interrupts are set but the counter remains at all-ones.

The MMC Transmit Interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>TXMCOLGPIS</th>
<th>TXSCOLGPIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Res</td>
<td>Res</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
</tr>
</tbody>
</table>
Bits 31:28  Reserved, must be kept at reset value.

- **Bit 27 TXLPITRCS**: MMC Transmit LPI transition counter interrupt status
  - This bit is set when the `Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)` counter reaches half of the maximum value or the maximum value.

- **Bit 26 TXLPIUSCIS**: MMC Transmit LPI microsecond counter interrupt status
  - This bit is set when the `Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)` counter reaches half of the maximum value or the maximum value.

Bits 25:22  Reserved, must be kept at reset value.

- **Bit 21 TXGPKTIS**: MMC Transmit Good Packet Counter Interrupt Status
  - This bit is set when the `Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)` counter reaches half of the maximum value or the maximum value.

Bits 20:16  Reserved, must be kept at reset value.

- **Bit 15 TXMCOLGPIS**: MMC Transmit Multiple Collision Good Packet Counter Interrupt Status
  - This bit is set when the `Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)` counter reaches half of the maximum value or the maximum value.

- **Bit 14 TXSCOLGPIS**: MMC Transmit Single Collision Good Packet Counter Interrupt Status
  - This bit is set when the `Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)` counter reaches half of the maximum value or the maximum value.

Bits 13:0  Reserved, must be kept at reset value.
MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C
Reset value: 0x0000 0000

The MMC Receive Interrupt Mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

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

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC Receive LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the **Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)** counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC Receive LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the **Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)** counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPM**: MMC Receive Unicast Good Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the **Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)** counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPM**: MMC Receive Alignment Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the **Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)** counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC Receive CRC Error Packet Counter Interrupt Mask
Setting this bit masks the interrupt when the **Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)** counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.
MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710
Reset value: 0x0000 0000

The MMC Transmit Interrupt Mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32-bit wide.

<table>
<thead>
<tr>
<th>Bit 31:0</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td>TXLPITRCIM: MMC Transmit LPI transition counter interrupt Mask</td>
</tr>
<tr>
<td></td>
<td>Setting this bit masks the interrupt when the Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR) counter reaches half of the maximum value or the maximum value.</td>
</tr>
<tr>
<td>Bit 26</td>
<td>TXLPIUSCIM: MMC Transmit LPI microsecond counter interrupt Mask</td>
</tr>
<tr>
<td></td>
<td>Setting this bit masks the interrupt when the Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR) counter reaches half of the maximum value or the maximum value.</td>
</tr>
<tr>
<td>Bit 21</td>
<td>TXGPKTIM: MMC Transmit Good Packet Counter Interrupt Mask</td>
</tr>
<tr>
<td></td>
<td>Setting this bit masks the interrupt when the Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD) counter reaches half of the maximum value or the maximum value.</td>
</tr>
<tr>
<td>Bit 15</td>
<td>TXMCOLGPIM: MMC Transmit Multiple Collision Good Packet Counter Interrupt Mask</td>
</tr>
<tr>
<td></td>
<td>Setting this bit masks the interrupt when the Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS) counter reaches half of the maximum value or the maximum value.</td>
</tr>
<tr>
<td>Bit 14</td>
<td>TXSCOLGPIM: MMC Transmit Single Collision Good Packet Counter Interrupt Mask</td>
</tr>
<tr>
<td></td>
<td>Setting this bit masks the interrupt when the Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS) counter reaches half of the maximum value or the maximum value.</td>
</tr>
<tr>
<td>Bit 13:0</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Tx single collision good packets register
(ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C
Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 TXSNGLCOLG[31:0]: Tx Single Collision Good Packets
This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

Tx multiple collision good packets register
(ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750
Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 TXMULTCOLG[31:0]: Tx Multiple Collision Good Packets
This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.
Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)

Address offset: 0x0768
Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Bits 31:0  **TXPKTG[31:0]**: Tx Packet Count Good
This field indicates the number of good packets transmitted.

Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)

Address offset: 0x0794
Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Bits 31:0  **RXCRCERR[31:0]**: Rx CRC Error Packets
This field indicates the number of packets received with CRC error.

Rx alignment error packets register
(ETH_RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798
Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

3492/3791  RM0477 Rev 6
Bits 31:0  **RXALGNERR[31:0]:** Rx Alignment Error Packets
This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

**Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)**
Address offset: 0x07C4
Reset value: 0x0000 0000
This register provides the number of good unicast packets received by Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **RXUCASTG[31:0]:** Rx Unicast Packets Good
This field indicates the number of good unicast packets received.

**Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)**
Address offset: 0x07EC
Reset value: 0x0000 0000
This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **TXLPIUSC[31:0]:** Tx LPI Microseconds Counter
This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.
Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0
Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  TXLPITRC[31:0]: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the LPI control and status register (ETH_MACLCSR)), the counter is incremented.

Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4
Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  RXLPITRC[31:0]: Rx LPI Transition counter

This field indicates the number of times Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8
Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**ETHernet (ETH): media access control (MAC) with DMA controller**

Bits 31:0  **RXLPITRC[31:0]:** Rx LPI Transition counter  
This field indicates the number of times Rx LPI Entry has occurred.

**L3 and L4 control 0 register (ETH_MACL3L4C0R)**

Address offset: 0x0900  
Reset value: 0x0000 0000  
The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>L4DPIM0</td>
<td>L4DPM0</td>
<td>L4SPIM0</td>
<td>L4SPM0</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>L3HDBM0[4:0]</td>
<td>L3HBSM0[4:0]</td>
<td>L3DAIM0</td>
<td>L3DAM0</td>
<td>L3SAIM0</td>
<td>L3SAM0</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

- **Bit 21**  **L4DPIM0:** Layer 4 Destination Port Inverse Match Enable  
  When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.  
  This bit is valid and applicable only when the L4DPM0 bit is set high.

- **Bit 20**  **L4DPM0:** Layer 4 Destination Port Match Enable  
  When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

- **Bit 19**  **L4SPIM0:** Layer 4 Source Port Inverse Match Enable  
  When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.  
  This bit is valid and applicable only when the L4SPM0 bit is set high.

- **Bit 18**  **L4SPM0:** Layer 4 Source Port Match Enable  
  When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

- **Bit 17**  Reserved, must be kept at reset value.

- **Bit 16**  **L4PEN0:** Layer 4 Protocol Enable  
  When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.  
  The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.
Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

- 0: No bits are masked.
- 1: LSB[0] is masked
- 2: Two LSBs [1:0] are masked
- ...
- 31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HDBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HDBM0 bits:

- 0: No bits are masked.
- 1: LSB[0] is masked
- 2: Two LSBs [1:0] are masked
- ...
- 31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

- 0: No bits are masked.
- 1: LSB[0] is masked
- 2: Two LSBs [1:0] are masked
- ...
- 31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

*Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.*

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.
Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

*Note:* When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

### Layer4 Address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td></td>
</tr>
</tbody>
</table>

**L4DP0[15:0]**

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
<td>nw</td>
</tr>
</tbody>
</table>

**L4SP0[15:0]**

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in **L3 and L4 control 0 register (ETH_MACL3L4C0R)**, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.
Layer3 Address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

Layer3 Address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

Bits 31:0 L3A00[31:0]: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 Address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.
Bits 31:0  **L3A10[31:0]**: Layer 3 Address 1 Field
When the L3PEN0 and L3SAM0 bits are set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.
When the L3PEN0 and L3DAM0 bits are set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.
When the L3PEN0 bit is reset and the L3SAM0 bit is set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 Address 2 filter 0 register (ETH_MACL3A20R)
Address offset: 0x0918
Reset value: 0x0000 0000
The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3A20[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **L3A20[31:0]**: Layer 3 Address 2 Field
When the L3PEN0 and L3SAM0 bits are set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.
When the L3PEN0 and L3DAM0 bits are set in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.
When the L3PEN0 bit is reset in the L3 and L4 control 0 register (ETH_MACL3L4C0R), this field is not used.

Layer3 Address 3 filter 0 register (ETH_MACL3A30R)
Address offset: 0x091C
Reset value: 0x0000 0000
The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3A30[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

|     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| L3A30[15:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
**L3 and L4 control 1 register (ETH_MACL3L4C1R)**

Address offset: 0x0930
Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

<table>
<thead>
<tr>
<th>Bit 31:22</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 21</td>
<td><strong>L4DPM1</strong>: Layer 4 Destination Port Match Enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.</td>
</tr>
<tr>
<td></td>
<td>This bit is valid and applicable only when the L4DPM1 bit is set high.</td>
</tr>
<tr>
<td>Bit 20</td>
<td><strong>L4DPM1</strong>: Layer 4 Destination Port Match Enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.</td>
</tr>
<tr>
<td>Bit 19</td>
<td><strong>L4SPIM1</strong>: Layer 4 Source Port Inverse Match Enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.</td>
</tr>
<tr>
<td></td>
<td>This bit is valid and applicable only when the L4SPIM1 bit is set high.</td>
</tr>
<tr>
<td>Bit 18</td>
<td><strong>L4SPIM1</strong>: Layer 4 Source Port Match Enable</td>
</tr>
<tr>
<td></td>
<td>When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.</td>
</tr>
<tr>
<td>Bit 17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
</tbody>
</table>
Bit 16  **L4PEN1**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11  **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match

<table>
<thead>
<tr>
<th>Condition: IPv4 packets</th>
<th>Condition: IPv6 packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:</td>
<td>Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:</td>
</tr>
<tr>
<td>0: No bits are masked.</td>
<td>0: No bits are masked.</td>
</tr>
<tr>
<td>1: LSb[0] is masked</td>
<td>1: LSb[0] is masked</td>
</tr>
<tr>
<td>2: Two LSbs [1:0] are masked</td>
<td>2: Two LSbs [1:0] are masked</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>31: All bits except MSb are masked.</td>
<td>127: All bits except MSb are masked.</td>
</tr>
</tbody>
</table>

This field is valid and applicable only when the L3DA1M or L3SAM1 bit is set.

Bits 10:6  **L3HSBM1[4:0]**: Layer 3 IP SA higher Bits Match

<table>
<thead>
<tr>
<th>Condition: IPv4 packets</th>
<th>Condition: IPv6 packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:</td>
<td>This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DA1M or L3SAM1 bit is set high.</td>
</tr>
<tr>
<td>0: No bits are masked.</td>
<td>0: No bits are masked.</td>
</tr>
<tr>
<td>1: LSb[0] is masked</td>
<td>1: LSb[0] is masked</td>
</tr>
<tr>
<td>2: Two LSbs [1:0] are masked</td>
<td>2: Two LSbs [1:0] are masked</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>31: All bits except MSb are masked.</td>
<td>127: All bits except MSb are masked.</td>
</tr>
</tbody>
</table>

Bit 5  **L3DA1M1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DA1M1 bit is set high.

Bit 4  **L3DA1M1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

*Note: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.*
Bit 3 L3SAIM1: Layer 3 IP SA Inverse Match Enable
When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching.
When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.
This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 L3SAM1: Layer 3 IP SA Match Enable
When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit
is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because
either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 L3PEN1: Layer 3 Protocol Enable
When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for
IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching
is enabled for IPv4 packets.
The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:16 L4DP1[15:0]: Layer 4 Destination Port Number Field
When the L4PEN1 bit is set and the L4DPM1 bit is set in the L3 and L4 control 1 register
(ETH_MACL3L4C1R), this field contains the value to be matched with the TCP Destination
Port Number field in the IPv4 or IPv6 packets.
When the L4PEN1 and L4DPM1 bits are set in L3 and L4 control 1 register
(ETH_MACL3L4C1R), this field contains the value to be matched with the UDP Destination
Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 L4SP1[15:0]: Layer 4 Source Port Number Field
When the L4PEN1 bit is set and the L4DPM1 bit is set in the L3 and L4 control 1 register
(ETH_MACL3L4C1R), this field contains the value to be matched with the TCP Source Port
Number field in the IPv4 or IPv6 packets.
When the L4PEN1 and L4DPM1 bits are set in L3 and L4 control 1 register
(ETH_MACL3L4C1R), this field contains the value to be matched with the UDP Source Port
Number field in the IPv4 or IPv6 packets.
Layer 3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A01[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 L3A01[31:0]: Layer 3 Address 0 Field

- When the L3PEN1 and L3SAM1 bits are set in the L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.
- When the L3PEN1 and L3DAM1 bits are set in the L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.
- When the L3PEN1 bit is reset and the L3SAM1 bit is set in the L3 and L4 control 1 register (ETH_MACL3L4C1R), this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer 3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944
Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>L3A11[31:16]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

| L3A11[15:0] | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |
Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

**Layer 3 address 2 filter 1 Register (ETH_MACL3A21R)**

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>L3A21[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw</td>
<td>rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

**Layer 3 address 3 filter 1 register (ETH_MACL3A31R)**

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>L3A31[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw</td>
<td>rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>L3A31[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw</td>
<td>rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:0  **L3A31[31:0]: Layer 3 Address 3 Field**

When the L3PEN1 and L3SAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the **L3 and L4 control 1 register (ETH_MACL3L4C1R)**, this field is not used.

**Timestamp control Register (ETH_MACTSCR)**

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the Receiver.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td><strong>TXTSSTSM</strong></td>
<td>Transmit Timestamp Status Mode</td>
</tr>
<tr>
<td>23</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td><strong>TSMSTREN</strong></td>
<td>Time Stamp Mode Enable</td>
</tr>
<tr>
<td>14</td>
<td><strong>TSEVTENA</strong></td>
<td>Time Stamp Event Enable</td>
</tr>
<tr>
<td>13</td>
<td>TSCPWEENA</td>
<td>Time Stamp Capture/Parallel Enable</td>
</tr>
<tr>
<td>12</td>
<td>TSPWENA</td>
<td>Time Stamp Parallel Enable</td>
</tr>
<tr>
<td>11</td>
<td><strong>SNAPTYPSEL[1:0]</strong></td>
<td>Enable network specific type parameter</td>
</tr>
<tr>
<td>10</td>
<td><strong>TSENMACADDR</strong></td>
<td>MAC Address Enable</td>
</tr>
<tr>
<td>9</td>
<td><strong>TSEVNTENA</strong></td>
<td>Time Stamp Event Enable</td>
</tr>
<tr>
<td>8</td>
<td><strong>TSIPV4ENATSIPV6ENA</strong></td>
<td>Time Stamp IPv4/IPv6 Event Enable</td>
</tr>
<tr>
<td>7</td>
<td>TSIPV4ENA</td>
<td>Time Stamp IPv4 Enable</td>
</tr>
<tr>
<td>6</td>
<td><strong>TSIPENA</strong></td>
<td>Time Stamp IPv6 Enable</td>
</tr>
<tr>
<td>5</td>
<td><strong>TSVER2ENA</strong></td>
<td>Time Stamp IPv6/Version 2 Event Enable</td>
</tr>
<tr>
<td>4</td>
<td><strong>TSENALL</strong></td>
<td>All Enable</td>
</tr>
<tr>
<td>3</td>
<td><strong>TSADDRREG</strong></td>
<td>Time Stamp Address Register</td>
</tr>
<tr>
<td>2</td>
<td><strong>TSUPDT</strong></td>
<td>Time Stamp Update Enable</td>
</tr>
<tr>
<td>1</td>
<td>TSINIT</td>
<td>Time Stamp Initial</td>
</tr>
<tr>
<td>0</td>
<td><strong>TSCFUPDT</strong></td>
<td>Time Stamp Capture/First Update Enable</td>
</tr>
</tbody>
</table>

Bit 28  **AV8021ASMEM**: AV 802.1AS Mode Enable

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When PTP offload feature is enabled, for the purpose of PTP offload, the transport specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25  Reserved, must be kept at reset value.

Bit 24  **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)** register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the **Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)**.

Bits 23:19  Reserved, must be kept at reset value.
Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering
When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.
When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.
For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.
For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTypSEL[1:0]**: Select PTP packets for Taking Snapshots
These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in Table 707: Timestamp Snapshot Dependency on ETH_MACTSCR bits.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master
When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages
When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see Table 707: Timestamp Snapshot Dependency on ETH_MACTSCR bits.

Bit 13 **TSIPv4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP
When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPv6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP
When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets
When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format
When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control
When this bit is set, the Timestamp Low register rolls over after 0x3B9A C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets
When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bits 7:6 Reserved, must be kept at reset value.
Bit 5 **TSADDR**: Update Addend Register
When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 **Reserved, must be kept at reset value.**

Bit 3 **TSUPDT**: Update Timestamp
When this bit is set, the system time is updated (added or subtracted) with the value specified in *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2 **TSINIT**: Initialize Timestamp
When this bit is set, the system time is initialized (overwritten) with the value specified in the *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update
When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp
When this bit is set, the timestamp is added for Transmit and Receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.
On the Receive side, the MAC processes the 1588 packets only if this bit is set.

**Subsecond increment register (ETH_MACSSIR)**
Address offset: 0x0B04
Reset value: 0x0000 0000

In Coarse Update mode (bit TSCFUPDT in *Timestamp control Register (ETH_MACTSCR)*), the value in this register is added to the system time every clock cycle of clk_ptp_ref_i. In Fine Update mode, the value in this register is added to the system time whenever the Accumulator gets an overflow.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SSINC[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:24  Reserved, must be kept at reset value.

Bits 23:16  **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in Timestamp control Register *(ETH_MACTSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0  Reserved, must be kept at reset value.
System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08
Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSS[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TSS[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **TSS[31:0]**: Timestamp Second
The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C
Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSSS[30:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TSSS[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds
The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in Timestamp control Register (ETH_MACTSCR), each bit represents 1 ns. The maximum value is 0x3B9A C9FF after which it rolls-over to zero.
System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10
Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in Timestamp control Register (ETH_MACTSCR).

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the seconds part of the update.

- When ADDSUB is reset, this field must be programmed with the seconds part of the update value.
- When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.

For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF FFFE (that is, $2^{32} - 2$).

**Caution:** When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0 in System time nanoseconds update register (ETH_MACSTNUR). The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in ~0.46 ns) even if 0 ns must be subtracted.

For example, to subtract 2.000000000 seconds from the system time, the TSS field in the System time seconds update register (ETH_MACSTSUR) must be 0xFFFF FFFE (that is, $2^{32} - 1$) and the System time nanoseconds update register (ETH_MACSTNUR) must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14
Reset value: 0x0000 0000

![System time nanoseconds update register (ETH_MACSTNUR)](image-url)
Bit 31  ADDSUB: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register. When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0  TSSS[30:0]: Timestamp subseconds

The value in this field is the subseconds part of the update.

- ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.
- ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the Timestamp control Register (ETH_MACTSCR).
- TSCTRLSSR field in the Timestamp control Register (ETH_MACTSCR) is 1:
  - The programmed value must be $10^5 - <\text{subsecond value}>$.
  - Each bit represents 1 ns and the programmed value should not exceed 0x3B9A C9FF.
- TSCTRLSSR field in the Timestamp control Register (ETH_MACTSCR) is 0:
  - The programmed value must be $2^{31} - <\text{subsecond value}>$.
  - Each bit represents an accuracy of 0.46 ns.

For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF FFFF (that is, $2^{31} - 1$), when TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR) is reset and 0x3B9A C9FF (that is, $10^5 - 1$), when TSCTRLSSR bit in Timestamp control Register (ETH_MACTSCR) is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.

For example, to subtract 2.000000000 seconds from the system time, System time nanoseconds update register (ETH_MACSTNUR) must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the System time seconds update register (ETH_MACSTSUR) must be 0xFFFF FFFE (that is, $2^{32} - 1$).

**Timestamp addend register (ETH_MACTSAR)**

Address offset: 0x0B18
Reset value: 0x0000 0000

This register value is used only when the system time is configured for Fine Update mode (TSCFUPD bit in the Timestamp control Register (ETH_MACTSCR)). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

<table>
<thead>
<tr>
<th>Bit 31:16</th>
<th>Bits 31:16</th>
<th>Bits 30:0</th>
<th>Bits 30:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TSAR[15:0]**

Bits 31:0  TSAR[31:0]: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.
**Timestamp status register (ETH_MACTSSR)**

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>rc_r</td>
<td></td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td>rc_r</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Bits 31:30 Reserved, must be kept at reset value.**

**Bits 29:25 ATSSN[4:0]: Number of Auxiliary Timestamp Snapshots**

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

**Bit 24 ATSTM: Auxiliary Timestamp Snapshot Trigger Missed**

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

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

**Bits 19:16 ATSTSN[3:0]: Auxiliary Timestamp Snapshot Trigger Identifier**

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

- Bit 16: Auxiliary trigger 0
- Bit 17: Auxiliary trigger 1
- Bit 18: Auxiliary trigger 2
- Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

**Bit 15 TXTSSIS: Tx Timestamp Status Interrupt Status**

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the *Tx timestamp status seconds register (ETH_MACTXTSSSR)* is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

**Bits 14:4 Reserved, must be kept at reset value.**
Bit 3 **TSTRGTER0:** Timestamp Target Time Error
This bit is set when the latest target time programmed in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR elapses (see **PPS target time seconds register (ETH_MACPPSTTSR)** and **PPS target time nanoseconds register (ETH_MACPPSTTNR)**). This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

Bit 2 **AUXSTSTRIG:** Auxiliary Timestamp Trigger Snapshot
This bit is set high when the auxiliary snapshot is written to the FIFO. This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

Bit 1 **TSTARGTO:** Timestamp Target Time Reached
When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTSR and ETH_MACPPSTTSNR registers (see **PPS target time seconds register (ETH_MACPPSTTSR)** and **PPS target time nanoseconds register (ETH_MACPPSTTNR)**). This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

Bit 0 **TSSOVF:** Timestamp Seconds Overflow
When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 0xFFFF FFFF. This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in **CSR software control register (ETH_MACCSRSWCR)** is set).

**Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)**
Address offset: 0x0B30
Reset value: 0x0000 0000
This register contains the nanosecond part of timestamp captured for Transmit packets when Tx status is disabled.

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXTSSMIS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 30:0</th>
<th>30:16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXTSSLO</td>
<td></td>
</tr>
<tr>
<td>rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f rc_f</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 **TXTSSMIS:** Transmit Timestamp Status Missed
When this bit is set, it indicates one of the following:
- The timestamp of the current packet is ignored if TXTSSTSM bit of the **Timestamp control Register (ETH_MACTSCR)** is reset
- The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the **Timestamp control Register (ETH_MACTSCR)** is set.

Bits 30:0 **TXTSSLO[30:0]:** Transmit Timestamp Status Low
This field contains the 31 bits of the Nanoseconds field of the Transmit packet’s captured timestamp.
**Tx timestamp status seconds register (ETH_MACTXTSSSR)**

Address offset: 0x0B34  
Reset value: 0x0000 0000  
The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXTSSHI[31:16]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High  
This field contains the lower 32 bits of the Seconds field of Transmit packet's captured timestamp.

**Auxiliary control register (ETH_MACACR)**

Address offset: 0x0B40  
Reset value: 0x0000 0000  
The Auxiliary Timestamp Control register controls the Auxiliary Timestamp snapshot.

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
```

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

- **Bit 7 ATSEN3**: Auxiliary Snapshot 3 Enable  
  - This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

- **Bit 6 ATSEN2**: Auxiliary Snapshot 2 Enable  
  - This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

- **Bit 5 ATSEN1**: Auxiliary Snapshot 1 Enable  
  - This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.
Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable
This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.
Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear
When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

**Auxiliary timestamp nanoseconds register (ETH_MACATSNR)**

Address offset: 0x0B48
Reset value: 0x0000 0000

The Auxiliary timestamp nanoseconds register (ETH_MACATSNR), along with Auxiliary timestamp seconds register (ETH_MACATSSR), gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in Timestamp status register (ETH_MACTSSR) indicate the fill-level of the FIFO. The top of the FIFO is removed only when Auxiliary timestamp seconds register (ETH_MACATSSR) is read.

<table>
<thead>
<tr>
<th></th>
<th>AUXTSLO[30:16]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

**Auxiliary timestamp seconds register (ETH_MACATSSR)**

Address offset: 0x0B4C
Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

<table>
<thead>
<tr>
<th></th>
<th>AUXTSLO[30:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 Reserved, must be kept at reset value.
Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp
Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

<table>
<thead>
<tr>
<th></th>
<th>AUXTSH[31:16]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

**Auxiliary timestamp seconds register (ETH_MACATSSR)**

Address offset: 0x0B4C
Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

<table>
<thead>
<tr>
<th></th>
<th>AUXTSH[15:0]</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>29</td>
<td>0</td>
<td>28</td>
<td>0</td>
<td>27</td>
<td>0</td>
<td>26</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>14</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp
Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

**Timestamp Ingress asymmetric correction register (ETH_MACTSIACR)**
Address offset: 0x0B50
Reset value: 0x0000 0000
The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction
This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000. The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

**Timestamp Egress asymmetric correction register (ETH_MACTSEACR)**
Address offset: 0x0B54
Reset value: 0x0000 0000
The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction
This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}. For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFDF 8000, which is the 2's complement of 0x0002 8000(2.5 × 2^{16}). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003 4CCC (3.3 × 2^{16}).
**Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)**

Address offset: 0x0B58  
Reset value: 0x0000 0000  
This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

Bits 31:0  **TSIC[31:0]: Timestamp Ingress Correction**  
This field contains the ingress path correction value as defined by the Ingress Correction expression.

**Timestamp Egress correction nanosecond register (ETH_MACTSECNR)**

Address offset: 0x0B5C  
Reset value: 0x0000 0000  
This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

Bits 31:0  **TSEC[31:0]: Timestamp Egress Correction**  
This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

**PPS control register (ETH_MACPPSCR)**

Address offset: 0x0B70  
Reset value: 0x0000 0000

---

**Table Examples**

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0B58</td>
<td>Timestamp Ingress correction nanosecond register (ETH_MACTSICNR)</td>
</tr>
<tr>
<td>0x0B5C</td>
<td>Timestamp Egress correction nanosecond register (ETH_MACTSECNR)</td>
</tr>
<tr>
<td>0x0B70</td>
<td>PPS control register (ETH_MACPPSCR)</td>
</tr>
</tbody>
</table>
Bits 31:7 Reserved, must be kept at reset value.

Bits 6:5 **TRGTMODSEL[1:0]**: Target Time Register Mode for PPS Output
This field indicates the Target Time registers *(PPS target time seconds register (ETH_MACPPSTTSR)) and PPS target time nanoseconds register (ETH_MACPPSTTNR)* mode for PPS output signal:
00: Target Time registers are programmed only for generating the interrupt event.
01: Reserved, must not be used
10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable
When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control
This field controls the frequency of the PPS output *(eth_ptp_pps_out)* signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:
0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.
0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.
0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.
1000: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

.. 1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

**Note:** In the binary rollover mode, the PPS output *(eth_ptp_pps_out)* has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- **When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.**
- **When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of**
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high).
- **When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of**
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high).

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.
PPS control register [alternate] (ETH_MACPPSCR)

Address offset: 0x0B70
Reset value: 0x0000 0000

Bits 31:7  Reserved, must be kept at reset value.

Bits 6:5  TRGTMODSEL0[1:0]: Target Time Register Mode for PPS Output
This field indicates the Target Time registers (PPS target time seconds register (ETH_MACPPSTTSR) and PPS target time nanoseconds register (ETH_MACPPSTTNR)) mode for PPS output signal:
00: Target Time registers are programmed only for generating the interrupt event.
01: Reserved, must not be used
10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4  PPSEN0: Flexible PPS Output Mode Enable
When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).
Bits 3:0 PPSCMD[3:0]: Flexible PPS Output (eth_ptp_pps_out) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are 'all zero'. The following list describes the values of PPSCMD0:

0000: No Command
0001: START Single Pulse.
This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.
0010: START Pulse Train.
This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.
0011: Cancel START.
This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.
0100: STOP Pulse Train at time.
This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.
0101: STOP Pulse Train immediately.
This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).
0110: Cancel STOP Pulse train.
This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.
0111 to 1111: Reserved, must not be used

PPS target time seconds register (ETH_MACPPSTTSR)

Address offset: 0x0B80
Reset value: 0x0000 0000

The PPS output Target Time Seconds register, along with PPS target time nanoseconds register (ETH_MACPPSTTNR), is used to schedule an interrupt event (Bit TSSOVF of Timestamp status register (ETH_MACTSSR)) when the system time exceeds the value programmed in these registers.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>TSTRH0[31:16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
<tr>
<td>TSTRH0[15:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 TSTRH0[31:0]: PPS Target Time Seconds Register

This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the PPS control register (ETH_MACPPSCR).
**PPS target time nanoseconds register (ETH_MACPPSTTNR)**

Address offset: 0x0B84
Reset value: 0x0000 0000

The PPS Target Time Nanoseconds register is present only when more than one Flexible PPS output is selected.

### PPS Target Time Nanoseconds register (ETH_MACPPSTTNR)

#### TRGTBUSY

- **Bit 31**: TRGTBUSY0: PPS Target Time Register Busy
  
  The MAC sets this bit when the PPSCMD0 field in the PPS control register (ETH_MACPPSCR) is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain. The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

#### TTSL0[30:0]

- **Bits 30:0**: TTSL0[30:0]: Target Time Low for PPS Register
  
  This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in PPS control register (ETH_MACPPSCR).

  When the TSCTRLSSR bit is set in the Timestamp control Register (ETH_MACTSCR), this value should not exceed 0x3B9A C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of subsecond increment value.

**PPS interval register (ETH_MACPPSIR)**

Address offset: 0x0B88
Reset value: 0x0000 0000

The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output.

### PPS Interval register (ETH_MACPPSIR)

#### PPSINT0[31:16]

#### PPSINT0[15:0]
Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

**PPS width register (ETH_MACPPSWR)**

Address offset: 0x0B8C
Reset value: 0x0000 0000

The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**PPSWIDTH0[31:16]**

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

*Note*: The value programmed in this register must be lesser than the value programmed in **PPS interval register (ETH_MACPPSIR)**.

**PTP Offload control register (ETH_MACPOCR)**

Address offset: 0x0BC0
Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**DN[7:0]**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**Note**: The value programmed in this register must be lesser than the value programmed in **PPS interval register (ETH_MACPPSIR)**.
Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]: Domain Number**
This field indicates the domain Number in which the PTP node is operating.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation
When this bit is set, the Delay Request and Delay response are not generated for received
SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger
When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically
released after the PTP Pdelay_Req message is transmitted. The application should set the
APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger
When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared
after the PTP SYNC message is transmitted. The application should set the ASYNCECN bit for
this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval
programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer
Transparent mode.

Bit 1 **ASYNCECN**: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval
programmed or trigger from application, when the MAC is programmed to be in Clock Master
mode.

Bit 0 **PTOEN**: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

**PTP Source Port Identity 0 Register (ETH_MACSPI0R)**

Address offset: 0x0BC4
Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

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

**SPI[31:16]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

**SPI[15:0]**

Bits 31:0 **SPI[31:0]: Source Port Identity 0**
This field indicates bits [31:0] of sourcePortIdentity of PTP node.
PTP Source port identity 1 register (ETH_MACSPI1R)
Address offset: 0x0BC8
Reset value: 0x0000 0000
This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)
Address offset: 0x0BCC
Reset value: 0x0000 0000
This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

Log message interval register (ETH_MACLMIR)
Address offset: 0x0BD0
Reset value: 0x0000 0000
This register contains the periodic intervals for automatic PTP packet generation.
Bits 31:24  **LMPDRI[7:0]**: Log Min Pdelay_Req Interval  
This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11  Reserved, must be kept at reset value.

Bits 10:8  **DRSYNCR[2:0]**:  
Delay_Req to SYNC Ratio  
In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.  
0: DelayReq generated for every received SYNC  
1: DelayReq generated every alternate reception of SYNC  
2: for every 4 SYNC messages  
3: for every 8 SYNC messages  
4: for every 16 SYNC messages  
5: for every 32 SYNC messages  
Others: Reserved, must not be used  
The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0  **LSI[7:0]**:  
Log Sync Interval  
This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.
# Ethernet register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>ETH_MACCR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SARC2[0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IPC</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IPG[2:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>GPSLGE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SAP</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ASR</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>VTR</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>JD</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>JE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>FES</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DM</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>LM</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ECOSED</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DCRS</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DR</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>BL[1:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DC</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DCLEN[1:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>TE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RE</td>
<td>0</td>
</tr>
<tr>
<td>0x0004</td>
<td>ETH_MACECR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>EIPSEN[4:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>EIPSEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>USP</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>DDBCC</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>GPSL[13:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x0008</td>
<td>ETH_MACPFR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IPF</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>VTFE</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>VTHM</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>EVLRXS</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>EVLS[1:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>EVLS[1:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>ERIVLT</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>EVLS[1:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x000C</td>
<td>ETH_MACWTR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>IPF</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT31T[31:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT63T[32:1:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x0010</td>
<td>ETH_MACT0R</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT31T[31:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT63T[32:1:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x0014</td>
<td>ETH_MACT1R</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT31T[31:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT63T[32:1:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x0018 - 0x004C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0050</td>
<td>ETH_MACVTR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT31T[31:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT63T[32:1:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x0054</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0058</td>
<td>ETH_MACVHTR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT31T[31:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT63T[32:1:0]</td>
<td>0</td>
</tr>
<tr>
<td>0x005C</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x0060</td>
<td>ETH_MACVIR</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td>ARPEN</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>RA</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT31T[31:0]</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>HT63T[32:1:0]</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 754. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>reset value</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0064</td>
<td>ETH_MACiVIR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0068 -</td>
<td>ETH_MACQTXFCR</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0070</td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0074 -</td>
<td>ETH_MACRXFCR</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0090</td>
<td>Reset value</td>
<td>0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0094-</td>
<td>ETH_MACISR</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B0</td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00B4</td>
<td>ETH_MACIER</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C0</td>
<td>ETH_MACPCSR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C4</td>
<td>ETH_MACRWKPFR</td>
<td>MACRWKPFR[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C8 -</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00CC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Offset value</td>
<td>Reset value</td>
<td>Offset value</td>
<td>Reset value</td>
<td>Offset value</td>
</tr>
<tr>
<td>---------</td>
<td>------------------------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>0x00D0</td>
<td>ETH_MACLCSR</td>
<td>31</td>
<td>0 0 0 0 0 0</td>
<td>30</td>
<td>0 0 0 0 0 0</td>
<td>29</td>
</tr>
<tr>
<td>0x00D4</td>
<td>ETH_MACLTCR</td>
<td>31</td>
<td>1 1 1 1 1 0</td>
<td>30</td>
<td>1 0 0 0 0 0</td>
<td>29</td>
</tr>
<tr>
<td>0x00D8</td>
<td>ETH_MACLETR</td>
<td>31</td>
<td>0 0 0 0 0 0</td>
<td>30</td>
<td>0 0 0 0 0 0</td>
<td>29</td>
</tr>
<tr>
<td>0x00DC</td>
<td>ETH_MAC1USTCR</td>
<td>31</td>
<td>0 0 0 0 0 0</td>
<td>30</td>
<td>0 0 0 0 0 0</td>
<td>29</td>
</tr>
<tr>
<td>0x00E0- 0x010C</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0110</td>
<td>ETH_MACVR</td>
<td>31</td>
<td>0x0000 0x3242</td>
<td>30</td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>0x0114</td>
<td>ETH_MACDR</td>
<td>31</td>
<td>0 0 0</td>
<td>30</td>
<td></td>
<td>29</td>
</tr>
<tr>
<td>0x0118</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x011C</td>
<td>ETH_MACHWF0R</td>
<td>31</td>
<td>0 0 0 0 0 0</td>
<td>30</td>
<td>0 0 0 0 0 0</td>
<td>29</td>
</tr>
<tr>
<td>0x0120</td>
<td>ETH_MACHWF1R</td>
<td>31</td>
<td>0 0 1 0 0 0</td>
<td>30</td>
<td>0 0 1 0 0 0</td>
<td>29</td>
</tr>
</tbody>
</table>

Table 754. Ethernet MAC register map and reset values (continued)
### Table 754. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name reset value</th>
<th>Offset</th>
<th>Register name reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0124</td>
<td>ETH_MACHWF2R</td>
<td>0x0128</td>
<td>ETH_MACHWF3R</td>
</tr>
<tr>
<td></td>
<td>0x012C - 0x01FC Reserved</td>
<td></td>
<td>0x01FC - Reserved</td>
</tr>
<tr>
<td>0x0200</td>
<td>ETH_MACMDIOAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0204</td>
<td>ETH_MACMDIODR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0210</td>
<td>ETH_MACARPAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0230</td>
<td>ETH_MACCSSRSW CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0300</td>
<td>ETH_MACA0HR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0304</td>
<td>ETH_MACA0LR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Register Descriptions
- **ETH_MACHWF2R**
- **ETH_MACHWF3R**
- **ETH_MACMDIOAR**
- **ETH_MACMDIODR**
- **ETH_MACARPAR**
- **ETH_MACCSSRSW CR**
- **ETH_MACA0HR**
- **ETH_MACA0LR**
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0308</td>
<td>ETH_MACA1HR</td>
<td>00000000 1111111111111111</td>
<td>0x03C</td>
<td>ETH_MACA1LR</td>
<td>11111111111111111111111111111111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0310</td>
<td>ETH_MACA2HR</td>
<td>00000000 1111111111111111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0314</td>
<td>ETH_MACA2LR</td>
<td>11111111111111111111111111111111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0318</td>
<td>ETH_MACA3HR</td>
<td>00000000 1111111111111111</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x031C</td>
<td>ETH_MACA3LR</td>
<td>11111111111111111111111111111111</td>
</tr>
<tr>
<td>0x0320-0x06FC</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0700</td>
<td>ETH_MMC_CONTROL</td>
<td>00000000 1111111111111111</td>
<td>0x0704</td>
<td>ETH_MMC_RX_INTERRUPT</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0708</td>
<td>ETH_MMC_TX_INTERRUPT</td>
<td>0000 0000 0000 0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Reset value</th>
<th>Offset</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>AE</td>
<td>30</td>
<td>SA</td>
</tr>
</tbody>
</table>

Table 754. Ethernet MAC register map and reset values (continued)
### Table 754. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x070C</td>
<td>ETH_MMC_RX_INTERRUPT_-MASK</td>
<td>31</td>
<td>RXPITRCIM</td>
<td>26</td>
<td>RXALPUSCM</td>
<td>21</td>
<td>RXUCGPM</td>
<td>16</td>
<td>RXAQNRRPM</td>
<td>11</td>
<td>RXCRCRPM</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>RXLPITRCIM</td>
<td></td>
<td>RXAQNRRPM</td>
<td></td>
<td>RXCRCRPM</td>
<td></td>
<td>RXCRCRPM</td>
<td></td>
<td>RXCRCRPM</td>
</tr>
<tr>
<td>0x0710</td>
<td>ETH_MMC_TX_INTERRUPT_-MASK</td>
<td>31</td>
<td>TXPITRCIM</td>
<td>26</td>
<td>TXAQNRRPM</td>
<td>21</td>
<td>TXUCGPM</td>
<td>16</td>
<td>TXCRCRPM</td>
<td>11</td>
<td>TXAQNRRPM</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>TXPITRCIM</td>
<td></td>
<td>TXAQNRRPM</td>
<td></td>
<td>TXUCGPM</td>
<td></td>
<td>TXCRCRPM</td>
<td></td>
<td>TXAQNRRPM</td>
</tr>
<tr>
<td>0x0714 - 0x0748</td>
<td>Reserved</td>
<td></td>
<td>0x074C ETH_TX_SINGLE_</td>
<td>31</td>
<td>TXSNGLCOLG[31:0]</td>
<td>26</td>
<td>TXSNGLCOLG[31:0]</td>
<td>21</td>
<td>TXSNGLCOLG[31:0]</td>
<td>16</td>
<td>TXSNGLCOLG[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>COLLISION_GOOD_PACKETS</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0750</td>
<td>ETH_TX_MULTIPLIC</td>
<td>31</td>
<td>TXMULTCOLG[31:0]</td>
<td>26</td>
<td>TXMULTCOLG[31:0]</td>
<td>21</td>
<td>TXMULTCOLG[31:0]</td>
<td>16</td>
<td>TXMULTCOLG[31:0]</td>
<td>11</td>
<td>TXMULTCOLG[31:0]</td>
</tr>
<tr>
<td></td>
<td>COLLISION_GOOD_PACKETS</td>
<td></td>
<td>TXMULTCOLG[31:0]</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0754 - 0x0764</td>
<td>Reserved</td>
<td></td>
<td>0x0768 ETH_TX_PACKET_</td>
<td>31</td>
<td>TXPKTG[31:0]</td>
<td>26</td>
<td>TXPKTG[31:0]</td>
<td>21</td>
<td>TXPKTG[31:0]</td>
<td>16</td>
<td>TXPKTG[31:0]</td>
</tr>
<tr>
<td></td>
<td>COUNT_GOOD</td>
<td></td>
<td>TXPKTG[31:0]</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x076C - 0x0790</td>
<td>Reserved</td>
<td></td>
<td>0x0794 ETH_RX_CRC_</td>
<td>31</td>
<td>RXCRCERR[31:0]</td>
<td>26</td>
<td>RXCRCERR[31:0]</td>
<td>21</td>
<td>RXCRCERR[31:0]</td>
<td>16</td>
<td>RXCRCERR[31:0]</td>
</tr>
<tr>
<td></td>
<td>ERROR_PACKETS</td>
<td></td>
<td>RXCRCERR[31:0]</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x0798</td>
<td>ETH_RX_ALIGNMENT_</td>
<td>31</td>
<td>RXALGNERR[31:0]</td>
<td>26</td>
<td>RXALGNERR[31:0]</td>
<td>21</td>
<td>RXALGNERR[31:0]</td>
<td>16</td>
<td>RXALGNERR[31:0]</td>
<td>11</td>
<td>RXALGNERR[31:0]</td>
</tr>
<tr>
<td></td>
<td>ERROR_PACKETS</td>
<td></td>
<td>RXALGNERR[31:0]</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x079C - 0x07C0</td>
<td>Reserved</td>
<td></td>
<td>0x07C4 ETH_RX_UNICAST_</td>
<td>31</td>
<td>RXUCASTG[31:0]</td>
<td>26</td>
<td>RXUCASTG[31:0]</td>
<td>21</td>
<td>RXUCASTG[31:0]</td>
<td>16</td>
<td>RXUCASTG[31:0]</td>
</tr>
<tr>
<td></td>
<td>PACKETS_GOOD</td>
<td></td>
<td>RXUCASTG[31:0]</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0x07C8 - 0x07E8</td>
<td>Reserved</td>
<td></td>
<td>0x07E8 Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
</tbody>
</table>
### Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x07EC</td>
<td>ETH_TX_LPI_</td>
<td>TXLPIUSC[31:0]</td>
</tr>
<tr>
<td></td>
<td>USEC_CNTR</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07F0</td>
<td>ETH_TX_LPI_</td>
<td>TXLPITRC[31:0]</td>
</tr>
<tr>
<td></td>
<td>TRAN_CNTR</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07F4</td>
<td>ETH_RX_LPI_</td>
<td>RXLPIUSC[31:0]</td>
</tr>
<tr>
<td></td>
<td>USEC_CNTR</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07F8</td>
<td>ETH_RX_LPI_</td>
<td>RXLPITRC[31:0]</td>
</tr>
<tr>
<td></td>
<td>TRAN_CNTR</td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x07FC-</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x08FC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0900</td>
<td>ETH_MACL3L4C0R</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0904</td>
<td>ETH_MACL4A0R</td>
<td>L4DP0[15:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0908-</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x090C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0910</td>
<td>ETH_MACL3A00R</td>
<td>L3A00[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0914</td>
<td>ETH_MACL3A10R</td>
<td>L3A10[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0918</td>
<td>ETH_MACL3A20R</td>
<td>L3A20[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x091C</td>
<td>ETH_MACL3A30R</td>
<td>L3A30[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x0920-</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>0x092C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0930</td>
<td>ETH_MACL3L4C1R</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value 0x00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
### Table 754. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Register bit field</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0934</td>
<td>ETH_MACL4A1R</td>
<td>L4DP[15:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>L4SP1[15:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0938 -</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x093C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0940</td>
<td>ETH_MACL3A01R</td>
<td>L3A01[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0944</td>
<td>ETH_MACL3A11R</td>
<td>L3A11[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0948</td>
<td>ETH_MACL3A21R</td>
<td>L3A21[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x094C</td>
<td>ETH_MACL3A31R</td>
<td>L3A31[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0950 -</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0AFC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0B00</td>
<td>ETH_MACTSCR</td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B04</td>
<td>ETH_MACSSIR</td>
<td>SSINC[7:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B08</td>
<td>ETH_MACSTR</td>
<td>TSS[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B0C</td>
<td>ETH_MACSTR</td>
<td>TSSS[30:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B10</td>
<td>ETH_MACSTSUR</td>
<td>TSS[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B14</td>
<td>ETH_MACSTNUR</td>
<td>TSSS[30:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B18</td>
<td>ETH_MACTSAR</td>
<td>TSAR[31:0]</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x0B1C</td>
<td>Reserved</td>
<td></td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>
Table 754. Ethernet MAC register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0B20</td>
<td>ETH_MACTSSR</td>
<td>0x0B28</td>
<td>Reserved</td>
<td>0x0B2C</td>
<td>Reserved</td>
<td>0x0B30</td>
<td>ETH_MACTXTSSNR</td>
<td>0x0B34</td>
<td>ETH_MACTXTSSSR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXTSSIS</td>
<td></td>
<td>TXTSSH[31:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0B38</td>
<td>ETH_MACACR</td>
<td>0x0B40</td>
<td>ETH_MACSNR</td>
<td>0x0B44</td>
<td>Reserved</td>
<td>0x0B48</td>
<td>ETH_MACSNR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x0B50</td>
<td>ETH_MACSIACR</td>
<td>0x0B54</td>
<td>ETH_MACSIEACR</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>Offset</td>
<td>Register name</td>
<td>Reset value</td>
<td>Offset</td>
<td>Register name</td>
<td>Reset value</td>
<td>Offset</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------</td>
<td>-------------</td>
<td>--------------</td>
<td>------------------------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>0x0B70</td>
<td>ETH_MACPPSCR</td>
<td></td>
<td>0x0B70</td>
<td>ETH_MACPPSCR (alternate)</td>
<td></td>
<td>0x0B74 - 0x0B7C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0B80</td>
<td>ETH_MACPSTTSR</td>
<td>TSTRH[31:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0B84</td>
<td>ETH_MACPSTTNR</td>
<td>TTSLO[30:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0B88</td>
<td>ETH_MACPSIR</td>
<td>PPSINT0[31:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0B8C</td>
<td>ETH_MACPSWR</td>
<td>PPSWIDTH0[31:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0B90 - 0x0BBC</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0BC0</td>
<td>ETH_MACOCR</td>
<td></td>
<td>0x0BC0</td>
<td>ETH_MACOCR</td>
<td></td>
<td>0x0BC4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0x0BC8</td>
<td>ETH_MACSP1R</td>
<td>SPI1[31:0]</td>
<td></td>
</tr>
</tbody>
</table>

Table 754. Ethernet MAC register map and reset values (continued)
Refer to \textit{Section 2.3 on page 149} for the register boundary addresses.
65  **HDMI-CEC controller (CEC)**

65.1  **HDMI-CEC introduction**

Consumer electronics control (CEC) is part of HDMI (high-definition multimedia interface) standard. It contains a protocol that provides high-level control functions between various audiovisual products. CEC operates at low speeds, with minimum processing and memory overhead.

The HDMI-CEC controller provides hardware support for this protocol.

65.2  **HDMI-CEC controller main features**

- Complies with HDMI-CEC v1.4 specification
- Independent 32 kHz CEC kernel (refer to section RCC kernel clock distribution)
- Works in Stop mode for ultra-low-power applications
- Configurable signal-free time before start of transmission
  - Automatic by hardware, according to CEC state and transmission history
  - Fixed by software (7 timing options)
- Configurable peripheral address (OAR)
- Supports Listen mode
  - Enables reception of CEC messages sent to destination address different from OAR without interfering with the CEC line
- Configurable Rx-tolerance margin
  - Standard tolerance
  - Extended tolerance
- Receive-error detection
  - Bit rising error (BRE), with optional stop of reception (BRESTP)
  - Short bit period error (SBPE)
  - Long bit period error (LBPE)
- Configurable error-bit generation
  - on BRE detection (BREGEN)
  - on LBPE detection (LBPEGEN)
  - always generated on SBPE detection
- Transmission error detection (TXERR)
- Arbitration lost detection (ARBLST)
  - with automatic transmission retry
- Transmission underrun detection (TXUDR)
- Reception overrun detection (RXOVR)
65.3 HDMI-CEC functional description

65.3.1 HDMI-CEC pin and internal signals

The CEC bus consists of a single bidirectional line that is used to transfer data in and out of the device. It is connected to a +3.3 V supply voltage via a 27 kΩ pull-up resistor. The output stage of the device must have an open-drain or open-collector to allow a wired-AND connection.

The HDMI-CEC controller manages the CEC bidirectional line as an alternate function of a standard GPIO, assuming that it is configured as alternate function open drain. The 27 kΩ pull-up must be added externally to the microcontroller.

To not interfere with the CEC bus when the application power is removed, it is mandatory to isolate the CEC pin from the bus in such conditions. This can be done by using a MOS transistor, as shown on Figure 989.

Table 756 lists the internal signals that are exchanged between the HDMI-CEC and other functional blocks (such as RCC and EXTI).

<table>
<thead>
<tr>
<th>Table 755. HDMI pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>---</td>
</tr>
</tbody>
</table>
| CEC | Bidirectional | Two states:  
– 1 = high impedance  
– 0 = low impedance  
A 27 kΩ resistor must be added externally. |

<table>
<thead>
<tr>
<th>Table 756. HDMI-CEC internal input/output signals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal name</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>cec_wkup</td>
</tr>
<tr>
<td>cec_it</td>
</tr>
<tr>
<td>cec_pclk</td>
</tr>
<tr>
<td>cec_ker_ck</td>
</tr>
</tbody>
</table>
65.3.2 **HDMI-CEC block diagram**

Figure 989. HDMI-CEC block diagram

65.3.3 **Message description**

All transactions on the CEC line consist of an initiator and one or more followers. The initiator is responsible for sending the message structure and the data. The follower is the recipient of any data and is responsible for setting any acknowledgment bits.

A message is conveyed in a single frame that consists of a start bit followed by a header block and optionally an opcode and a variable number of operand blocks.

All these blocks are made of a 8-bit payload - most significant bit is transmitted first - followed by an end of message (EOM) bit and an acknowledge (ACK) bit.

The EOM bit is set in the last block of a message and kept reset in all others. In case a message contains additional blocks after an EOM is indicated, those additional blocks must be ignored. The EOM bit may be set in the header block to ‘ping’ other devices, to make sure they are active.

The acknowledge bit is always set to high impedance by the initiator so that it can be driven low either by the follower that has read its own address in the header, or by the follower that needs to reject a broadcast message.

The header consists of the source logical address field, and the destination logical address field. Note that the special address 0xF is used for broadcast messages.

Figure 990. Message structure
65.3.4 Bit timing

The format of the start bit is unique and identifies the start of a message. It must be validated by its low duration and its total duration.

All remaining data bits in the message, after the start bit, have consistent timing. The high-to-low transition at the end of the data bit is the start of the next data bit except for the final bit where the CEC line remains high.

65.4 Arbitration

All devices transmitting - or retransmitting - a message onto the CEC line must ensure that it has been inactive for a number of bit periods. This signal-free time is defined as the time...
starting from the final bit of the previous frame and depends on the initiating device and the current status as shown in the figure below.

**Figure 993. Signal free time**

![Figure 993. Signal free time](MS31007V1)

Since only one initiator is allowed at any one time, an arbitration mechanism is provided to avoid conflict when more than one initiator begins transmitting at the same time.

CEC line arbitration starts with the leading edge of the start bit and continues until the end of the initiator address bits within the header block. During this period, the initiator must monitor the CEC line, if whilst driving the line to high impedance it reads it back to 0. Assuming then it has lost arbitration, it stops transmitting and becomes a follower.

**Figure 994. Arbitration phase**

![Figure 994. Arbitration phase](MS31008V1)

**Figure 995** shows an example for a SFT of three nominal bit periods.

**Figure 995. SFT of three nominal bit periods**

![Figure 995. SFT of three nominal bit periods](MS31009V1)

A configurable time window is counted before starting the transmission.

In the SFT = 0 configuration, HDMI-CEC performs automatic SFT calculation ensuring compliance with the HDMI-CEC standard:

- 2.5 data bit periods if the CEC is the last bus initiator with unsuccessful transmission
- 4 data bit periods if the CEC is the new bus initiator
- 6 data bit periods if the CEC is the last bus initiator with successful transmission
This is done to guarantee the maximum priority to a failed transmission and the lowest one to the last initiator that completed successfully its transmission.

Otherwise there is the possibility to configure the SFT bits to count a fixed timing value. Possible values are 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5 data bit periods.

65.4.1 SFT option bit

In case of SFTOPT = 0 configuration, SFT starts being counted when the start-of-transmission command is set by software (TXSOM = 1).

In case of SFTOPT = 1, SFT starts automatically being counted by the HDMI-CEC device when a bus-idle or line error condition is detected. If the SFT timer is completed at the time TXSOM command is set then transmission starts immediately without latency. If the SFT timer is still running instead, the system waits until the timer elapses before transmission can start.

In case of SFTOPT = 1 a bus-event condition starting the SFT timer is detected in the following cases:

- In case of a regular end of transmission/reception, when TXEND/RXEND bits are set at the minimum nominal data bit duration of the last bit in the message (ACK bit).
- In case of a transmission error detection, SFT timer starts when the TXERR transmission error is detected (TXERR = 1).
- In case of a missing acknowledge from the CEC follower, the SFT timer starts when the TXACKE bit is set, that is at the nominal sampling time of the ACK bit.
- In case of a transmission underrun error, the SFT timer starts when the TXUDR bit is set at the end of the ACK bit.
- In case of a receive error detection implying reception abort, the SFT timer starts at the same time the error is detected. If an error bit is generated, then SFT starts being counted at the end of the error bit.
- In case of a wrong start bit or of any uncoded low impedance bus state from idle, the SFT timer is restarted as soon as the bus comes back to hi-impedance idleness.

65.5 Error handling

65.5.1 Bit error

If a data bit - excluding the start bit - is considered invalid, the follower is expected to notify such error by generating a low bit period on the CEC line of 1.4 to 1.6 times the nominal data bit period (3.6 ms nominally).

![Figure 996. Error bit timing](MS31010V1)
65.5.2 Message error

A message is considered lost and therefore may be retransmitted under the following conditions:

- a message is not acknowledged in a directly addressed message
- a message is negatively acknowledged in a broadcast message
- a low impedance is detected on the CEC line while it is not expected (line error)

Three kinds of error flag can be detected when the CEC interface is receiving a data bit:

65.5.3 Bit rising error (BRE)

BRE (bit rising error) is set when a bit rising edge is detected outside the windows where it is expected (see Figure 997). BRE flag also generates a CEC interrupt if the BREIE = 1.

In the case of a BRE detection, the message reception can be stopped according to the BRESTP bit value and an error bit can be generated if BREGEN bit is set.

When BRE is detected in a broadcast message with BRESTP = 1 an error bit is generated even if BREGEN = 0 to enforce initiator's retry of the failed transmission. Error bit generation can be disabled by configuring BREGEN = 0, BRDNOGEN = 1.

65.5.4 Short bit period error (SBPE)

SBPE is set when a bit falling edge is detected earlier than expected (see Figure 997). SBPE flag also generates a CEC interrupt if the SBPEIE = 1.

An error bit is always generated on the line in case of a SBPE error detection. An error bit is not generated upon SBPE detection only when Listen mode is set (LSTN = 1) and the following conditions are met:

- A directly addressed message is received containing SBPE
- A broadcast message is received containing SBPE AND BRDNOGEN = 1

65.5.5 Long bit period error (LBPE)

LBPE is set when a bit falling edge is not detected in a valid window (see Figure 997). LBPE flag also generates a CEC interrupt if the LBPEIE = 1.

LBPE always stops the reception, an error bit is generated on the line when LBPEGEN bit is set.

When LBPE is detected in a broadcast message an error bit is generated even if LBPEGEN = 0 to enforce initiator’s retry of the failed transmission. Error bit generation can be disabled by configuring LBPEGEN = 0, BRDNOGEN = 1.

Note: The BREGEN = 1, BRESTP = 0 configuration must be avoided.
Table 757. Error handling timing parameters

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_s$</td>
<td>x</td>
<td>0</td>
<td>Bit start event.</td>
</tr>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0.3</td>
<td>The earliest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$T_{n1}$</td>
<td>x</td>
<td>0.6</td>
<td>The nominal time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>0.8</td>
<td>The latest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>$T_{ns}$</td>
<td>x</td>
<td>1.05</td>
<td>Nominal sampling time.</td>
</tr>
<tr>
<td>$T_3$</td>
<td>1</td>
<td>1.2</td>
<td>The earliest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>$T_{n0}$</td>
<td>x</td>
<td>1.5</td>
<td>The nominal time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>1.7</td>
<td>The latest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>$T_5$</td>
<td>1</td>
<td>1.85</td>
<td>The earliest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.05</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>2.4</td>
<td>The nominal data bit period.</td>
</tr>
<tr>
<td>$T_6$</td>
<td>0</td>
<td>2.75</td>
<td>The latest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.95</td>
<td></td>
</tr>
</tbody>
</table>
65.5.6 Transmission error detection (TXERR)

The CEC initiator sets the TXERR flag if detecting low impedance on the CEC line when it is transmitting high impedance and is not expecting a follower asserted bit. TXERR flag also generates a CEC interrupt if the TXERRIE = 1.

TXERR assertion stops the message transmission. Application is in charge to retry the failed transmission up to five times.

TXERR checks are performed differently depending on the different states of the CEC line and on the RX tolerance configuration.

**Figure 998. TXERR detection**

![TXERR detection diagram](image)

**Table 758. TXERR timing parameters**

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_s$</td>
<td>x</td>
<td>0</td>
<td>Bit start event.</td>
</tr>
<tr>
<td>$T_1$</td>
<td>1</td>
<td>0.3</td>
<td>The earliest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>0.6</td>
<td>The nominal time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td>$T_2$</td>
<td>0</td>
<td>0.8</td>
<td>The latest time for a low - high transition when indicating a logical 1.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>$T_{ns}$</td>
<td>x</td>
<td>1.05</td>
<td>Nominal sampling time.</td>
</tr>
<tr>
<td>$T_3$</td>
<td>1</td>
<td>1.2</td>
<td>The earliest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1.3</td>
<td></td>
</tr>
</tbody>
</table>
HDMI-CEC controller (CEC)

65.6 HDMI-CEC interrupts

An interrupt can be produced:
- during reception if a receive block transfer is finished or if a receive error occurs.
- during transmission if a transmit block transfer is finished or if a transmit error occurs.

Table 758. TXERR timing parameters (continued)

<table>
<thead>
<tr>
<th>Time</th>
<th>RXTOL</th>
<th>ms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{n0}$</td>
<td>x</td>
<td>1.5</td>
<td>The nominal time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td>$T_4$</td>
<td>0</td>
<td>1.7</td>
<td>The latest time a device is permitted return to a high impedance state (logical 0).</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>$T_5$</td>
<td>1</td>
<td>1.85</td>
<td>The earliest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>2.05</td>
<td></td>
</tr>
<tr>
<td>$T_{nf}$</td>
<td>x</td>
<td>2.4</td>
<td>The nominal data bit period.</td>
</tr>
<tr>
<td>$T_6$</td>
<td>0</td>
<td>2.75</td>
<td>The latest time for the start of a following bit.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2.95</td>
<td></td>
</tr>
</tbody>
</table>

Table 759. HDMI-CEC interrupts

<table>
<thead>
<tr>
<th>Interrupt event</th>
<th>Event flag</th>
<th>Enable control bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rx-byte received</td>
<td>RXBR</td>
<td>RXBRIE</td>
</tr>
<tr>
<td>End of reception</td>
<td>RXEND</td>
<td>RXENDIE</td>
</tr>
<tr>
<td>Rx-overrun</td>
<td>RXOVR</td>
<td>RXOVRIE</td>
</tr>
<tr>
<td>Rxbit rising error</td>
<td>BRE</td>
<td>BREIE</td>
</tr>
<tr>
<td>Rx-short bit period error</td>
<td>SBPE</td>
<td>SBPEIE</td>
</tr>
<tr>
<td>Rx-long bit period error</td>
<td>LBPE</td>
<td>LBPEIE</td>
</tr>
<tr>
<td>Rx-missing acknowledge error</td>
<td>RXACKE</td>
<td>RXACKEIE</td>
</tr>
<tr>
<td>Arbitration lost</td>
<td>ARBLST</td>
<td>ARBLSTIE</td>
</tr>
<tr>
<td>Tx-byte request</td>
<td>TXBR</td>
<td>TXBRIE</td>
</tr>
<tr>
<td>End of transmission</td>
<td>TXEND</td>
<td>TXENDIE</td>
</tr>
<tr>
<td>Tx-buffer underrun</td>
<td>TXUDR</td>
<td>TXUDRIE</td>
</tr>
<tr>
<td>Tx-error</td>
<td>TXERR</td>
<td>TXERRIE</td>
</tr>
<tr>
<td>Tx-missing acknowledge error</td>
<td>TXACKE</td>
<td>TXACKEIE</td>
</tr>
</tbody>
</table>
65.7  HDMI-CEC registers

Refer to Section 1.2 for a list of abbreviations used in register descriptions. The registers have to be accessed by words (32 bits).

65.7.1  CEC control register (CEC_CR)

Address offset: 0x00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TXSOM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CECEN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rs</td>
<td>rs</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 2  TXEOM: Tx end of message
The TXEOM bit is set by software to command transmission of the last byte of a CEC message. TXEOM is cleared by hardware at the same time and under the same conditions as for TXSOM.

0: TXDR data byte is transmitted with EOM = 0.
1: TXDR data byte is transmitted with EOM = 1.

Note: TXEOM must be set when CECEN = 1.
TXEOM must be set before writing transmission data to TXDR.
If TXEOM is set when TXSOM = 0, transmitted message consists of 1 byte (HEADER) only (PING message).

Bit 1  TXSOM: Tx start of message
TXSOM is set by software to command transmission of the first byte of a CEC message. If the CEC message consists of only one byte, TXEOM must be set before of TXSOM.
Start-bit is effectively started on the CEC line after SFT is counted. If TXSOM is set while a message reception is ongoing, transmission starts after the end of reception.
TXSOM is cleared by hardware after the last byte of the message is sent with a positive acknowledge (TXEND = 1), in case of transmission underrun (TXUDR = 1), negative acknowledge (TXACKE = 1), and transmission error (TXERR = 1). It is also cleared by CECEN = 0. It is not cleared and transmission is automatically retried in case of arbitration lost (ARBLST = 1).
TXSOM can be also used as a status bit informing application whether any transmission request is pending or under execution. The application can abort a transmission request at any time by clearing the CECEN bit.

0: No CEC transmission is on-going.
1: CEC transmission command

Note: TXSOM must be set when CECEN = 1.
TXSOM must be set when transmission data is available into TXDR.
HEADER first four bits containing own peripheral address are taken from TXDR[7:4], not from CEC_CFRGR.OAR that is used only for reception.
Bit 0 **CECEN**: CEC enable

The CECEN bit is set and cleared by software. CECEN = 1 starts message reception and enables the TXSOM control. CECEN = 0 disables the CEC peripheral, clears all bits of CEC_CR register and aborts any on-going reception or transmission.

0: CEC peripheral is off.
1: CEC peripheral is on.

### 65.7.2 CEC configuration register (CEC_CFGR)

This register is used to configure the HDMI-CEC controller.

- **Address offset**: 0x04
- **Reset value**: 0x0000 0000

**Caution**: It is mandatory to write CEC_CFGR only when CECEN = 0.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>LSTN</td>
<td>Listen mode</td>
</tr>
<tr>
<td></td>
<td>OAR[14:0]</td>
<td>Own addresses configuration</td>
</tr>
<tr>
<td>8</td>
<td>SFTOP</td>
<td>SFT option bit</td>
</tr>
<tr>
<td>7</td>
<td>OGEN</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>LPPEG</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BREGE</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>BREST</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>RXTOL</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SFT[2:0]</td>
<td></td>
</tr>
</tbody>
</table>

**Bit 31 LSTN**: Listen mode

LSTN bit is set and cleared by software.

0: CEC peripheral receives only message addressed to its own address (OAR). Messages addressed to different destination are ignored. Broadcast messages are always received.

1: CEC peripheral receives messages addressed to its own address (OAR) with positive acknowledge. Messages addressed to different destination are received, but without interfering with the CEC bus: no acknowledge sent.

**Bits 30:16 OAR[14:0]: Own addresses configuration**

The OAR bits are set by software to select which destination logical addresses has to be considered in receive mode. Each bit, when set, enables the CEC logical address identified by the given bit position.

At the end of HEADER reception, the received destination address is compared with the enabled addresses. In case of matching address, the incoming message is acknowledged and received. In case of non-matching address, the incoming message is received only in listen mode (LSTN = 1), but without acknowledge sent. Broadcast messages are always received.

Example:

OAR = 0b0000 0000 0010 0001 means that CEC acknowledges addresses 0x0 and 0x5. Consequently, each message directed to one of these addresses is received.

**Bits 15:9** Reserved, must be kept at reset value.

**Bit 8 SFTOP**: SFT option bit

The SFTOPT bit is set and cleared by software.

0: SFT timer starts when TXSOM is set by software.
1: SFT timer starts automatically at the end of message transmission/reception.
Bit 7  **BRDNOGEN**: Avoid error-bit generation in broadcast
The BRDNOGEN bit is set and cleared by software.
0: BRE detection with BRESTP = 1 and BREGEN = 0 on a broadcast message generates an error-bit on the CEC line. LBPE detection with LBPEGEN = 0 on a broadcast message generates an error-bit on the CEC line.
1: Error-bit is not generated in the same condition as above. An error-bit is not generated even in case of an SBPE detection in a broadcast message if listen mode is set.

Bit 6  **LBPEGEN**: Generate error-bit on long bit period error
The LBPEGEN bit is set and cleared by software.
0: LBPE detection does not generate an error-bit on the CEC line.
1: LBPE detection generates an error-bit on the CEC line.

Note: **If BRDNOGEN = 0, an error-bit is generated upon LBPE detection in broadcast even if LBPEGEN = 0.**

Bit 5  **BREGEN**: Generate error-bit on bit rising error
The BREGEN bit is set and cleared by software.
0: BRE detection does not generate an error-bit on the CEC line.
1: BRE detection generates an error-bit on the CEC line (if BRESTP is set).

Note: **If BRDNOGEN = 0, an error-bit is generated upon BRE detection with BRESTP = 1 in broadcast even if BREGEN = 0.**

Bit 4  **BRESTP**: Rx-stop on bit rising error
The BRESTP bit is set and cleared by software.
0: BRE detection does not stop reception of the CEC message. Data bit is sampled at 1.05 ms.
1: BRE detection stops message reception.

Bit 3  **RXTOL**: Rx-tolerance
The RXTOL bit is set and cleared by software.
0: Standard tolerance margin:
  – Start-bit, +/- 200 µs rise, +/- 200 µs fall
  – Data-bit: +/- 200 µs rise. +/- 350 µs fall
1: Extended tolerance
  – Start-bit: +/- 400 µs rise, +/- 400 µs fall
  – Data-bit: +/- 300 µs rise, +/- 500 µs fall

Bits 2:0  **SFT[2:0]**: Signal free time
SFT bits are set by software. In the SFT = 0x0 configuration, the number of nominal data bit periods waited before transmission is ruled by hardware according to the transmission history. In all the other configurations the SFT number is determined by software.
0x0
  – 2.5 data-bit periods if CEC is the last bus initiator with unsuccessful transmission (ARBLST = 1, TXERR = 1, TXUDR = 1 or TXACKE = 1)
  – 4 data-bit periods if CEC is the new bus initiator
  – 6 data-bit periods if CEC is the last bus initiator with successful transmission (TXEOM = 1)
0x1: 0.5 nominal data bit periods
0x2: 1.5 nominal data bit periods
0x3: 2.5 nominal data bit periods
0x4: 3.5 nominal data bit periods
0x5: 4.5 nominal data bit periods
0x6: 5.5 nominal data bit periods
0x7: 6.5 nominal data bit periods
### 65.7.3 CEC Tx data register (CEC_TXDR)

Address offset: 0x8  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved, must be kept at reset value.</td>
<td>TXD[7:0]</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **TXD[7:0]:** Tx data  
TXD is a write-only register containing the data byte to be transmitted.

### 65.7.4 CEC Rx data register (CEC_RXDR)

Address offset: 0xC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved, must be kept at reset value.</td>
<td>RXD[7:0]</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **RXD[7:0]:** Rx data  
RXD is read-only and contains the last data byte that has been received from the CEC line.

### 65.7.5 CEC interrupt and status register (CEC_ISR)

Address offset: 0x10  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-13</th>
<th>Bit 12-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved, must be kept at reset value.</td>
<td>TXACK E TXERR TXUDR TXEND TXBR ARBLS T RXACK E LBPE SBPE BRE RXOV R RXEND RXBR</td>
</tr>
</tbody>
</table>

Bits 31:13 Reserved, must be kept at reset value.
Bit 12 **TXACKE:** Tx-missing acknowledge error
In transmission mode, TXACKE is set by hardware to inform application that no acknowledge was received. In case of broadcast transmission, TXACKE informs application that a negative acknowledge was received. TXACKE aborts message transmission and clears TXSOM and TXEOM controls.
TXACKE is cleared by software write at 1.

Bit 11 **TXERR:** Tx-error
In transmission mode, TXERR is set by hardware if the CEC initiator detects low impedance on the CEC line while it is released. TXERR aborts message transmission and clears TXSOM and TXEOM controls.
TXERR is cleared by software write at 1.

Bit 10 **TXUDR:** Tx-buffer underrun
In transmission mode, TXUDR is set by hardware if application was not in time to load TXDR before of next byte transmission. TXUDR aborts message transmission and clears TXSOM and TXEOM control bits.
TXUDR is cleared by software write at 1

Bit 9 **TXEND:** End of transmission
TXEND is set by hardware to inform application that the last byte of the CEC message has been successfully transmitted. TXEND clears the TXSOM and TXEOM control bits.
TXEND is cleared by software write at 1.

Bit 8 **TXBR:** Tx-byte request
TXBR is set by hardware to inform application that the next transmission data has to be written to TXDR. TXBR is set when the 4th bit of currently transmitted byte is sent. Application must write the next byte to TXDR within six nominal data-bit periods before transmission underrun error occurs (TXUDR).
TXBR is cleared by software write at 1.

Bit 7 **ARBLST:** Arbitration lost
ARBLST is set by hardware to inform application that CEC device is switching to reception due to arbitration lost event following the TXSOM command. ARBLST can be due either to a contending CEC device starting earlier or starting at the same time but with higher HEADER priority. After ARBLST assertion TXSOM bit keeps pending for next transmission attempt.
ARBLST is cleared by software write at 1.

Bit 6 **RXACKE:** Rx-missing acknowledge
In receive mode, RXACKE is set by hardware to inform application that no acknowledgment was seen on the CEC line. RXACKE applies only for broadcast messages and in listen mode also for not directly addressed messages (destination address not enabled in OAR). RXACKE aborts message reception.
RXACKE is cleared by software write at 1.

Bit 5 **LBPE:** Rx-long bit period error
LBPE is set by hardware in case a data-bit waveform is detected with long bit period error. LBPE is set at the end of the maximum bit-extension tolerance allowed by RXTOL, in case falling edge is still longing. LBPE always stops reception of the CEC message. LBPE generates an error-bit on the CEC line if LBPEGEN = 1. In case of broadcast, error-bit is generated even in case of LBPEGEN = 0.
LBPE is cleared by software write at 1.

Bit 4 **SBPE:** Rx-short bit period error
SBPE is set by hardware in case a data-bit waveform is detected with short bit period error. SBPE is set at the time the anticipated falling edge occurs. SBPE generates an error-bit on the CEC line.
SBPE is cleared by software write at 1.
Bit 3 BRE: Rx-bit rising error
BRE is set by hardware in case a data-bit waveform is detected with bit rising error. BRE is set either at the time the misplaced rising edge occurs, or at the end of the maximum BRE tolerance allowed by RXTOL. In case rising edge is still longing, BRE stops message reception if BRESTP = 1. BRE generates an error-bit on the CEC line if BREGEN = 1.
BRE is cleared by software write at 1.

Bit 2 RXOVR: Rx-overrun
RXOVR is set by hardware if RXBR is not yet cleared at the time a new byte is received on the CEC line and stored into RXD. RXOVR assertion stops message reception so that no acknowledge is sent. In case of broadcast, a negative acknowledge is sent.
RXOVR is cleared by software write at 1.

Bit 1 RXEND: End of reception
RXEND is set by hardware to inform application that the last byte of a CEC message is received from the CEC line and stored into the RXD buffer. RXEND is set at the same time of RXBR.
RXEND is cleared by software write at 1.

Bit 0 RXBR: Rx-byte received
The RXBR bit is set by hardware to inform application that a new byte has been received from the CEC line and stored into the RXD buffer.
RXBR is cleared by software write at 1.

65.7.6 CEC interrupt enable register (CEC_IER)
Address offset: 0x14
Reset value: 0x0000 0000

Caution: It is mandatory to write CEC_IER only when CECEN = 0.

<table>
<thead>
<tr>
<th>Bit 31:13 Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

Bit 12 TXACKIE: Tx-missing acknowledge error interrupt enable
The TXACKIE bit is set and cleared by software.
0: TXACKE interrupt disabled
1: TXACKE interrupt enabled

Bit 11 TXERRIE: Tx-error interrupt enable
The TXERRIE bit is set and cleared by software.
0: TXERR interrupt disabled
1: TXERR interrupt enabled

Bit 10 TXUDRIE: Tx-underrun interrupt enable
The TXUDRIE bit is set and cleared by software.
0: TXUDR interrupt disabled
1: TXUDR interrupt enabled
Bit 9 **TXENDIE**: Tx-end of message interrupt enable
   The TXENDIE bit is set and cleared by software.
   0: TXEND interrupt disabled
   1: TXEND interrupt enabled

Bit 8 **TXBRIE**: Tx-byte request interrupt enable
   The TXBRIE bit is set and cleared by software.
   0: TXBR interrupt disabled
   1: TXBR interrupt enabled

Bit 7 **ARBLSTIE**: Arbitration lost interrupt enable
   The ARBLSTIE bit is set and cleared by software.
   0: ARBLST interrupt disabled
   1: ARBLST interrupt enabled

Bit 6 **RXACKIE**: Rx-missing acknowledge error interrupt enable
   The RXACKIE bit is set and cleared by software.
   0: RXACKE interrupt disabled
   1: RXACKE interrupt enabled

Bit 5 **LBPEIE**: Long bit period error interrupt enable
   The LBPEIE bit is set and cleared by software.
   0: LBPE interrupt disabled
   1: LBPE interrupt enabled

Bit 4 **SBPEIE**: Short bit period error interrupt enable
   The SBPEIE bit is set and cleared by software.
   0: SBPE interrupt disabled
   1: SBPE interrupt enabled

Bit 3 **BREIE**: Bit rising error interrupt enable
   The BREIE bit is set and cleared by software.
   0: BRE interrupt disabled
   1: BRE interrupt enabled

Bit 2 **RXOVRIE**: Rx-buffer overrun interrupt enable
   The RXOVRIE bit is set and cleared by software.
   0: RXOVR interrupt disabled
   1: RXOVR interrupt enabled

Bit 1 **RXENDIE**: End of reception interrupt enable
   The RXENDIE bit is set and cleared by software.
   0: RXEND interrupt disabled
   1: RXEND interrupt enabled

Bit 0 **RXBRIE**: Rx-byte received interrupt enable
   The RXBRIE bit is set and cleared by software.
   0: RXBR interrupt disabled
   1: RXBR interrupt enabled
## 65.7.7 HDMI-CEC register map

### Table 760. HDMI-CEC register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | CEC_CR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x04   | CEC_CFG      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |
| 0x08   | CEC_TXDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |
| 0x0C   | CEC_RXDR     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |
| 0x10   | CEC_ISR      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |
| 0x14   | CEC_IER      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0 |

Refer to Section 2.3 for the register boundary addresses.
66 Debug infrastructure

66.1 Introduction

The debug infrastructure allows software designers to debug and trace their embedded software.

The debug features can be controlled via a JTAG/Serial-wire debug access port, using industry standard debugging tools. A trace port allows data to be captured for logging and analysis.

The trace and debug system is designed to support a variety of typical use cases:

- **Low cost trace**
  Limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt detection and program counter sampling. Single-wire trace can be maintained even when the processor is switched off or clock-stopped.

- **Breakpoint debugging**
  The processor can be debugged using equipment connected to the JTAG/SWD debug port. This allows breakpoint and watchpoint setting, code stepping, memory access and so on.

- **Tracing code execution via the trace port**
  Trace information is combined into a single trace stream and output to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.

- **Continuous trace capturing in a circular buffer**
  Instead of streaming it off-chip, the combined trace information can be stored on-chip in a circular buffer. The trace storage can be started and stopped by different means such as a debugger command, a software command, an external trigger signal, an internal event, and so on.

- **Draining the buffer to the trace port**
  The stored trace can be dumped off-chip to the trace port analyzer. The buffer draining can be initiated by the debugger, software, external trigger, internal event and so on.

- **Reading the buffer with the debugger**
  The debugger can read the contents of the trace buffer via the debug port. This is slower than the trace port, but allows basic trace functionality on the debugger without the cost of a trace port analyzer.

- **Analyzing stored trace in software**
  The trace buffer can be read by the processor core, or transferred into system memory by DMA. This powerful feature allows built-in test software to monitor code execution in real time, analyze and identify faults, handle exceptions autonomously, and so on.

- **Uploading stored trace**
  The stored trace can also be uploaded to a host machine using one of the MCU’s many communication interfaces (USB, USART, SPI, I2C, Ethernet, CAN and so on). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.
66.2 Debug infrastructure features

A comprehensive set of trace and debug features is provided to support software development and system integration:

- Breakpoint debugging
- Code execution tracing
- Software instrumentation
- Cross-triggering
- JTAG debug port
- Serial-wire debug port
- Trigger input and output
- Serial-wire trace port
- Trace port
- Arm® CoreSight™ debug and trace components.

The CoreSight components are described at high level in this document. Detailed information is available in the Arm® documents referenced in Section 66.10.

66.3 Debug infrastructure functional description

66.3.1 Debug infrastructure block diagram

The block diagram shows the logical partitioning of the debug infrastructure.

![Block diagram of debug infrastructure](image-url)
66.3.2 Debug infrastructure powering, clocking and reset

Power domains

The debug components are all located in the core power domain. This means that a debug session is disconnected if the core is powered down, for example in standby mode. This can be prevented by enabling low power emulation mode via the DBGMCU_CR.DBG_STANDBY register bit.

Clock domains

The debugger supplies the clock for the debug port, swtclk, via the debug interface pin, JTCK/SWCLK. This clock is used to register the serial input data in both serial wire and JTAG mode, as well as to operate the state machines and internal logic of the debug port. It must therefore continue to toggle for several cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the ck_dbg domain, which covers the rest of the SWJ-DP and the access ports.

ck_dbg clocks the trace components in the core power domain: System ROM table 2, CoreSight trace funnel, ETF, system CTI and TPIU. It is a gated version of the core domain system clock (rcc_hclk3).

traceclkin is the trace port clock. It is derived from the TRACECLKIN input.
The debug clock, ck_dbg, can be enabled and disabled by a register bit in the DBGMCU or by the debugger using the CDBGWRUPREQ bit in the debug port CTRLSTAT register. The clock must be enabled before the debugger can access any of the debug features on the device. It should be disabled at power up and when the debugger is disconnected, to avoid wasting energy.

The debug and trace components included in the processor (ETM ITM, DWG, FPB and so on) are clocked with the corresponding core clock (rcc_c_ck).

**Debug with low-power modes**

The device includes power-saving features allowing individual power domains to be switched off or stopped when not required. If a power domain is switched off or not clocked, all debug components in that domain are inaccessible to the debugger. To avoid this, power saving mode emulation is implemented. If the emulation is enabled for a domain, the domain still enters power saving mode, but its clock and power are maintained. In other words, the domain behaves as if it is in power saving mode, while the debugger does not lose the connection.

The emulation mode is programmed in the MCU Debug (DBGMCU) unit. For more information, refer to Section 66.8

**Reset of debug infrastructure**

The debug components, except for the debug port and access ports, are reset by a system or power on reset. The debug port (SWJ-DP) is reset by a power-on reset or a low level on nJTRST.

**66.3.3 Security**

The trace and debug components allow a high degree of access to the processor and system during product development. In order to protect user code and ensure that the debug features can not be used to alter or compromise the normal operation of the finished product, these features can be disabled or limited in scope.

Debugger access is disabled while the processor is booting from system flash memory.

The following authentication signals are used by the system to determine which debug features are enabled or disabled:

- **dbgen**: global enable for all debug features
  
  0: All debug features are disabled.
  
  1: Debug features are enabled.

- **niden**: enables trace and performance monitoring (non-invasive debug).
  
  0: Trace generation is disabled.
  
  1: Trace generation is enabled.

For detailed information on the behavior of each component according to the state of the authentication signals, refer to the relevant component chapter or to the relevant Arm® technical documentation.
The state of the signals are set according to the debug state as shown in Table 761.

Table 761. Authentication signal states

<table>
<thead>
<tr>
<th>Debug state</th>
<th>Authentication signal state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>dbgen = 1 niden = 1</td>
<td>Debug and trace is enabled whatever the state of the processor.</td>
</tr>
<tr>
<td>CLOSED</td>
<td>dbgen = 0 niden = 0</td>
<td>Debug and trace is disabled.</td>
</tr>
</tbody>
</table>

The debug state depends on the product life cycle state, and the debug authentication state (see Section 66.3.4: Debug authentication).

Table 762. Authentication signal states

<table>
<thead>
<tr>
<th>Product life cycle state</th>
<th>Debug state</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>OPEN</td>
</tr>
<tr>
<td>CLOSED (debug not authenticated)</td>
<td>CLOSED</td>
</tr>
<tr>
<td>CLOSED (debug constrained(1))</td>
<td>OPEN</td>
</tr>
<tr>
<td>LOCKED</td>
<td>CLOSED</td>
</tr>
</tbody>
</table>

1. The level of debug access is defined by the SBS debug control register (SBS_DBGCR).
66.3.4 Debug authentication

Figure 1001. Product life cycle states and debug authentication states.

If the device is in CLOSED (product life cycle) state, the debug state is CLOSED. The debug authentication procedure allows a trusted debugger to reopen access without compromising sensitive information called the Root Of Trust (ROT).

Re-opening the debug is possible only if sensitive asset security is ensured. This is called Constrained Debug, as constraints ensure the security of the ROT information.

Alternatively, a partial or full regression mechanism can be used when security of sensitive information cannot be guaranteed. This is called regression, as regression ensures removal of the sensitive information before reopening the debug.

- Full regression corresponds to releasing all code and assets. The intermediate state which allows full regression management is called Regression.

Debug authentication control principle

The debug authentication feature is one of the most critical security features of the system considering that with a debugger one can get access to a large part of the system.

In order to control re-opening of the debug, the device imposes a debug authentication protocol.

The protocol implements a challenge response mechanism based on asymmetric cryptography to authenticate the host. It relies on a key pair, with a Public Key stored in the device, and a Private Key from the host library which is used to sign a random value (the challenge) generated by the device.
The protocol implements a bidirectional communication between the host and the device through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox. For details on this protocol, refer to the Application.

The Debug Authentication protocol is launched on a Power On Reset of the device, when an “open request” message is posted by the host.

The protocol is based on:
- Initial message => Posted by the host combined with a reset to launch the Debug Authentication process on the device.
- Challenge message => The device generates a random value, to be signed by the host, when sending back the response.
- Response => The host sends a message to the device proving its authenticity. This is done using a tool to generate a token.

The implementation of the feature on the device is ensured by code embedded in the System Flash. This code is called automatically after reset if an initial message has been posted by the host in the mailbox.

After a first sequence of mutual authentication to align on protocol version, type of device, etc. the device generates a random value that should be signed by the host with a Private Key when building the response.

The STMicroelectronics implementation is based on the ARM PSA-ADAC solution for Debug Authentication.

The Debug Authentication can be implemented using a proprietary or open source protocol. As this feature is critical for security, the STM32H7 comes with Debug Authentication provisioned in System Flash. STMicroelectronics provides the host tools integrated with some debug environments (IDEs).

### 66.4 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

The port can be configured as:
- a 5-pin standard JTAG debug port (JTAG-DP)
- a 2-pin (clock + data) “serial-wire” debug port (SW-DP)

The two modes are mutually exclusive, since they share the same IO pins.

By default, the JTAG-DP is selected on system or power-on reset. The five IOs are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on the JTDI, JTMS/SWDIO, and nJTRST lines, as well as a pull-down resistor on the JTCK/SWCLK line.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO: ....(50 or more ones),...,0,1,1,1,0,0,1,1,1,0,0,1,1,1... (50 or more ‘1’s)... JTCK/SWCLK must be cycled for each data bit.
In SW-DP mode, the unused JTAG lines JTDI, JTDO and nJTRST can be used for other
functions.

All SWJ port IOs can be reconfigured to other functions by software, in which case
debugging is no longer possible.

### 66.4.1 Serial wire debug port

The serial wire debug protocol uses two pins:
- **SWCLK**: clock from host to target
- **SWDIO**: bi-directional serial data

Serial data is transferred LSB first, synchronously with the clock. A transfer comprises three
phases:
1. packet request (8 bits) transmitted by the host
2. acknowledge response (3 bits) transmitted by the target
3. data transfer (33 bits) transmitted by the host (in the case of a write) or target (in the
case of a read)

The data transfer only occurs if the acknowledge response is OK.

If the direction of the data is reversed between each phase, a single clock cycle turn-around
time is inserted.

#### Table 763. Packet request

<table>
<thead>
<tr>
<th>Field bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Start</td>
<td>Must be “1”</td>
</tr>
<tr>
<td>1</td>
<td>APnDP</td>
<td>0: DP register access - see Table 767 for a list of DP registers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: AP register access - see Section 66.5</td>
</tr>
<tr>
<td>2</td>
<td>RnW</td>
<td>0: Write request</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Read request</td>
</tr>
<tr>
<td>4:3</td>
<td>A(3:2)</td>
<td>Address field of the DP or AP register (refer to Table 767 and Table 769)</td>
</tr>
<tr>
<td>5</td>
<td>Parity</td>
<td>Single bit parity of preceding bits</td>
</tr>
<tr>
<td>6</td>
<td>Stop</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Park</td>
<td>Not driven by host. Must be read as “1” by target.</td>
</tr>
</tbody>
</table>

#### Table 764. ACK response

<table>
<thead>
<tr>
<th>Field bits</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2:0</td>
<td>ACK</td>
<td>000b: Fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010b: Wait</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100b: OK</td>
</tr>
</tbody>
</table>
**Table 765. Data transfer**

<table>
<thead>
<tr>
<th>Bit field</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>WDATA or RDATA</td>
<td>Write or Read data</td>
</tr>
<tr>
<td>32</td>
<td>Parity</td>
<td>Single bit parity of 32 data bits</td>
</tr>
</tbody>
</table>

*Figure 1002 shows successful write and read transfers.*

*Figure 1002. SWD successful data transfer*

For any FAULT or WAIT ACK response from the target, the data transfer phase is canceled, unless overrun detection is enabled, in which case the data is ignored by the target (in the case of a write), or not driven (in the case of a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists of 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the Serial Wire debug protocol, refer to the Arm® Debug Interface Architecture Specification [1].

**Note:** The SWJ-DP implements SWD protocol version 2.
66.4.2  JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE 1149.1-1990. The state machine is shown in Figure 1003. It controls two scan chains, one associated with an instruction register (IR) and one with a number of data registers (DR).

When the TAPSM goes through the Capture-IR state, 0b0001 is transferred into the instruction register (IR) scan chain. The IR scan chain is connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each rising edge of JTCK. This means that on the first tick:
- The LSB of the IR scan chain is output on JTDO
- Bit [n] of the IR scan chain is transferred to bit [n-1]
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned in the IR scan chain is transferred to the instruction register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one of the data registers onto one of the DR scan chains, connected between JTDI and JTDO.
The value held in the instruction register determines which data register, and associated DR scan chain, is selected.

This data is then shifted while the TAPSM is in the Shift-DR state, in the same way as the IR shift in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned in the DR scan chain is transferred to the selected data register.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the nJTRST signal resets the state machine asynchronously to the Test-Logic-Reset state.

The data registers corresponding to the 4-bit IR instructions are listed in Table 766.

### Table 766. JTAG-DP data registers

<table>
<thead>
<tr>
<th>Instruction register</th>
<th>Data register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000 to 0111 (BYPASS)</td>
<td>(BYPASS)</td>
<td>1</td>
<td>Not implemented: BYPASS selected</td>
</tr>
<tr>
<td>1000</td>
<td>ABORT</td>
<td>35</td>
<td>Abort register</td>
</tr>
<tr>
<td>1001 (BYPASS)</td>
<td></td>
<td>1</td>
<td>Reserved: BYPASS selected</td>
</tr>
<tr>
<td>1010</td>
<td>DPACC</td>
<td>35</td>
<td>Debug port access register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Initiates the debug port and allows access to a debug port register.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– When transferring data IN:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bit 0 = RnW = Read request (1) or write request (0).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>– When transferring data OUT:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>010b = OK/fault</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>001b = Wait</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other = reserved</td>
</tr>
</tbody>
</table>
### Table 766. JTAG-DP data registers (continued)

<table>
<thead>
<tr>
<th>Instruction register</th>
<th>Data register</th>
<th>Scan chain length</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1011                 | APACC         | 35                | Access port access register initiates an access port and allows access to an access port register.  
  – When transferring data IN:  
    Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request  
    Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register.  
    Bit 0 = RnW = Read request (1) or write request (0).  
  – When transferring data OUT:  
    Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request  
    Bits 2:0 = ACK[2:0] = 3-bit Acknowledge:  
      010b = OK/Fault  
      001b = Wait  
      Other = reserved |
| 1100 (BYPASS)       | 1             | Reserved: BYPASS selected |
| 1101 (BYPASS)       | 1             | Reserved: BYPASS selected |
| 1110                 | IDCODE        | 32                | ID Code  
  0x6BA00477: Arm® JTAG debug port ID code |
| 1111                 | BYPASS        | 1                 | Bypass  
  A single JTCK cycle delay is inserted between JTDI and JTDO |

The DR registers are described in more detail in the Arm® Debug Interface Architecture Specification [1].
### 66.4.3 Debug port registers

The SW-DP and JTAG-DP both access the debug port (DP) registers. These are listed in Table 767.

The debugger can access the DP registers as follows:

1. Program the SELECT register DPBANKSEL field in the DP to select the register bank to be accessed (see Table 767).
2. Program the A[3:2] field in the DPACC register, if using JTAG, with the register address within the bank. Program the R/W bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A[3:2] and R/W fields are part of the Packet Request word sent to the SW-DP with the APnDP bit reset (see Table 763). The write data is sent in the data phase.

#### Table 767. Debug port registers

<table>
<thead>
<tr>
<th>Address</th>
<th>A[3:2] field value</th>
<th>R/W</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>00</td>
<td>R</td>
<td>DP_DPIDR register(^2). Contains the IDCODE for the debug port.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>DP_ABORT register(^1). aborts the current AP transaction. This register is also used to clear the error flags in the DP_CTRL/STAT register.</td>
</tr>
<tr>
<td>0x4</td>
<td>01</td>
<td>R/W</td>
<td>If DPBANKSEL[3:0] = 0x0 (DP_SELECT register): CTRLSTAT register. Controls the DP and provides status information. If DPBANKSEL[3:0] = 0x1 (DP_SELECT register): DP_DLCR register(^2). Controls the operating mode of the SWD Data Link. If DPBANKSEL[3:0] = 0x2 (DP_SELECT register): DP_TARGETID register. Provides target identification information. If DPBANKSEL[3:0] = 0x3 (DP_SELECT register): DLPIDR register(^2). Provides the SWD protocol version.</td>
</tr>
<tr>
<td>0x8</td>
<td>10</td>
<td>R</td>
<td>RESEND register(^2). Returns the value that was returned by the last AP read or DP_RDBUFF read, used in the event of a corrupted read transfer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>DP_SELECT register. Selects the access port, access port register bank, and DP register at address 0x4.</td>
</tr>
<tr>
<td>0xC</td>
<td>11</td>
<td>R</td>
<td>DP_RDBUFF register. Via JTAG-DP, enables the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation). Via SW-DP, contains the result of the preceding AP read access, allowing a new AP access to be avoided.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W</td>
<td>DP_TARGETSEL register(^2). On a write to DP_TARGETSEL immediately following a line reset sequence, the target is selected if the following conditions are both met: – Bits [31:28] match bits [31:28] in the DP_DLPIDR register. – Bits [27:0] match bits [27:0] in the DP_TARGETID register. Writing any other value deselects the target. Debug tools must write 0xFFFFFFFF to deselect all targets. This is an invalid DP_TARGETID value. All other invalid DP_TARGETID values are reserved.</td>
</tr>
</tbody>
</table>

---

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is "reserved" via JTAG-DP.
Debug port identification register (DP_DPIDR)

Address offset: 0x0
Reset value: 0x6BA0 2477

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:28 **REVISION[3:0]**: Revision code
0x6

Bits 27:20 **PARTNO[7:0]**: Debug port part number
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm®

Bit 0 Reserved, must be kept at reset value.

Debug port abort register (DP_ABORT)

Address offset: 0x0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ORUNERRCLR</th>
<th>WDEERRCLR</th>
<th>STKERRCLR</th>
<th>STKCMPCLR</th>
<th>DAPABORT</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>
Bits 31:5  Reserved, must be kept at reset value.

Bit 4  **ORUNERRCLR**: Overrun error clear bit
   0: No effect
   1: Clear CTRLSTAT register’s STICKYORUN bit

Bit 3  **WDERRCLR**: Write data error clear bit
   0: No effect
   1: Clear CTRLSTAT register’s WDATAERR bit

Bit 2  **STKERRCLR**: Sticky error clear bit
   0: No effect
   1: Clear CTRLSTAT register’s STICKYERR bit

Bit 1  **STKCMPCLR**: Sticky compare clear bit
   0: No effect
   1: Clear CTRLSTAT register’s STICKYCMP bit

Bit 0  **DAPABORT**: Abort current AP transaction
   The transaction is aborted if an excessive number of WAIT responses are returned, indicating that the transaction has stalled.
   0: No effect
   1: Abort transaction
Debug port control/status register (DP_CTRLSTAT)

Address offset: 0x4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25:24</th>
<th>Bit 23:12</th>
<th>Bit 11:8</th>
<th>Bit 9:8</th>
<th>Bit 7:4</th>
<th>Bit 3:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bit 31 **CSYSWRUPACK**: System domain power-up status bit - not used in this device

Bit 30 **CSYSWRUPREQ**: System domain power-up control bit - not used in this device

Bit 29 **CDBGWRUPACK**: Debug domain power-up status bit
This bit is read-only. It returns the status of the debug domain power-up acknowledge signal from the power controller.
0: domain powered down
1: domain powered up

Bit 28 **CDBGWRUPREQ**: Debug domain power-up/down control bit
This bit controls the debug domain power-up/down request signal to the power controller.
0: power-down requested
1: power-up requested

Bit 27 **CDBGRSSTACK**: Debug domain reset status bit - not used in this device

Bit 26 **CDBGRSRQ**: Debug domain reset control bit - not used in this device

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]**: Transaction counter
To program a sequence of transactions to incremental addresses via an AP, TRNCNT bits are loaded with the number of transactions to perform. It is decremented on successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: Pushed-compare and pushed-verify masking bits
The field indicates the bytes to be masked in pushed-compare and pushed-verify operations (DP_CTRLSTAT register’s field TRNMODE = 1 or 2). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.
0b1XXX: include byte lane 3 in comparisons
0bX1XX: include byte lane 2 in comparisons
0bXX1X: include byte lane 1 in comparisons
0bXXXX: include byte lane 0 in comparisons
Bit 7 **WDATAERR**: Write data error in SW-DP
The bit indicates:
- a parity or a framing error on the data phase of a write operation, or
- a write operation that had been accepted by the DP has then been discarded without being submitted to the AP
This bit is read-only. It is reset by writing 1 to the WDERRCLR bit of the DP_ABORT register.
0: No error
1: Error has occurred
This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response in SW-DP
This bit indicates the response to the last AP read access. It is read-only.
0: Read not OK
1: Read OK
This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read-only in SW-DP, R/W in JTAG-DP)
This bit indicates that an error occurred during an AP transaction.
0: No error
1: Error has occurred
In the SW-DP, this bit is reset by writing 1 to the STKERRCLR bit of the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.

Bit 4 **STICKYCMP**: Compare match (read-only in SW-DP, R/W in JTAG-DP)
This bit indicates that a match occurred in a pushed operation.
0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2
1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2
In the SW-DP, this bit is reset by writing 1 to the STKCMPCLR bit in the DP_ABORT register.
In the JTAG-DP, this bit is reset by programming it to 1.

Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations
For read operations, this field must be set to 0x0.
0x0: Normal operation - AP transactions are passed directly to the AP.
0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read operation is compared with the stored data. If they do not match, the STICKYCMP bit is set.
0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data. If they match, the STICKYCMP bit is set.
0x3: Reserved
In pushed operations, only the data bytes indicated by the MASKLANE field are included in the comparison.

Bit 1 **STICKYORUN**: Overrun (read-only in SW-DP, R/W in JTAG-DP)
This bit indicates that an overrun occurred (a new transaction received before previous transaction completed). This bit is only set if the ORUNDETECT bit is set.
0: No overrun
1: Overrun occurred
In the SW-DP, this bit is reset by writing 1 to the ABORT register's ORUNERRCLR bit. In the JTAG-DP, this bit is reset by writing a 1 to it.

Bit 0 **ORUNDETECT**: Overrun detection mode enable
0: Overrun detection disabled
1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until the STICKYORUN bit is cleared.
Debug port data link control register (DP_DLCR)

Address offset: 0x4
Reset value: 0x0000 0040

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

**Bits 9:8** \text{TURNROUND}[1:0]: Tristate period for SWDIO
- 0x0: 1 data bit period
- 0x1: 2 data bit periods
- 0x2: 3 data bit periods
- 0x3: 4 data bit periods

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

Debug port target identification register (DP_TARGETID)

Address offset: 0x4
Reset value: 0x1485 0041

**Bits 31:28** \text{TREVISION}[3:0]: Device revision number
- 0x1

**Bits 27:12** \text{TPARTNO}[15:0]: Target part number
- 0x4850: STM32H7Rx/7Sx

**Bits 11:1** \text{TDESIGNER}[10:0]: Target designer JEDEC code
- 0x020: STMicroelectronics

**Bit 0** Reserved, must be kept at reset value.
Debug port data link protocol identification register (DP_DLPIDR)

Address offset: 0x4
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>TINSTANCE[3:0]</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:28  **TINSTANCE[3:0]**: Target instance number
These bits define the instance number for this device in a multi-drop system.
0x0

Bits 27:4  Reserved, must be kept at reset value.

Bits 3:0  **PROTSVN[3:0]**: Serial Wire Debug protocol version
0x1: Version 2

Debug port resend register (DP_RESEND)

Address offset: 0x8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESEND[31:16]</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESEND[15:0]</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>RESEND[15:0]</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
</tbody>
</table>

Bits 31:0  **RESEND[31:0]**: Last AP read or DP RDBUFF read value
These bits contain the value that was returned by the last AP read or DP RDBUFF read.
Used in the event of a corrupted read transfer.
Debug port access port select register (DP_SELECT)

Address offset: 0x8
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

Bits 31:24  **APSEL[7:0]:** Access port select bits
These bits select the access port for the next transaction.
- 0x00: AP0 - Cortex-M7 debug access port (AHB-AP)
- 0x01: AP1 - D3 access port (AHB-AP)
- 0x02: AP2 - System debug access port (APB-AP)
- 0x03 to 0x1F: Reserved

Bits 23:8  Reserved, must be kept at reset value.

Bits 7:4  **APBANKSEL[3:0]:** AP register bank select bits
These bits select the 4-word register bank on the active AP for the next transaction.

Bits 3:0  **DPBANKSEL[3:0]:** DP register bank select bits
These bits select the register at address 0x4 of the debug port.
- 0x0: CTRLSTAT register
- 0x1: DLCR register
- 0x2: TARGETID register
- 0x3: DLPIDR register
- 0x4 to 0xF: Reserved

Debug port read buffer register (DP_RDBUFF)

Address offset: 0xC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>RDBUFF[31:16]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0  **RDBUFF[31:0]:** Last AP read value
The field contains the value returned by the last AP read access. There are two ways to retrieve the value returned by an AP read access:
- perform a second read access to the same address, which initiates a new transaction on the corresponding bus, or
- read the value returned by the last AP read access from the DP_RDBUFF register, in which case no new AP transaction occurs
## Debug port target identification register (DP_TARGETSEL)

Address offset: 0xC

Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>w w w w</td>
<td>w w w w w w w</td>
<td>w w w w w w w</td>
<td>w w w w w w w</td>
<td>w w w w</td>
</tr>
<tr>
<td>15:10</td>
<td>15 14 13 12</td>
<td>11 10 9 8</td>
<td>7 6 5 4</td>
<td>3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>11:1</td>
<td>TPARTNO[3:0]</td>
<td>TDESIGNER[10:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>w w w w</td>
<td>w w w w w w w</td>
<td>w w w w w w w</td>
<td>w w w w w w w</td>
<td>w w w w</td>
</tr>
</tbody>
</table>

- **Bits 31:28 TINSTANCE[3:0]:** Target instance number
  The field defines the instance number for the target device in a multi-drop system. It must be programmed with the same value as TINSTANCE field of DP_DLPIHDR register, in order to select this device.

- **Bits 27:12 TPARTNO[15:0]:** Target part number
  The field defines the part number for the target device. It must be programmed with the same value as TPARTNO field of DP_TARGETID register, in order to select this device.

- **Bits 11:1 TDESIGNER[10:0]:** Target designer JEDEC code
  The field defines the JEDEC code for the target device. It must be programmed with the same value as TDESIGNER field of DP_TARGETID register, in order to select this device.

- **Bit 0 Reserved, must be kept at reset value.**
### 66.4.4 Debug port register map

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interface.

The debug port address offset is 4 bits wide, where the 2 most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The 2 least significant bits are 00.

**Table 768. Debug port register map and reset values**

| Offset | Register name       | 31  | 30  | 29  | 28  | 27  | 26  | 25  | 24  | 23  | 22  | 21  | 20  | 19  | 18  | 17  | 16  | 15  | 14  | 13  | 12  | 11  | 10  | 9   | 8   | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|--------|--------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0x0    | DP_DPIDR           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0x0    | DP_ABORT           |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xd(1)| DP_CTRLSTAT        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xd(2)| DP_DLCR            |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0xd(3)| DP_TARGETID        |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0xd(4)| DP_DLPIDR          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x8    | DP_RESEND          |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |     |
|        |                    | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
Access ports (AP) attached to the DP:

1. **AP1**: Cortex-M7 access port (AHB-AP). Allows access to the debug and trace features integrated in the Cortex-M7 processor core via an AHB-Lite bus connected to the AHB port of the processor.

2. **AP0**: System access port (APB-AP). Allows access to the debug and trace features on the system APB debug bus, that is, all components not included in the processor core.

All access ports are of MEM-AP type, that is, the debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus. The AP registers are listed in Table 769.
The address of the AP registers is composed of:
- bits [7:4]: content of the DP_SELECT register’s APBANKSEL field
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see Table 766) or of the SW-DP Packet Request (see Table 763), depending on the debug interface used
- bits [1:0]: Always set to 0

The content of the SELECT register APSEL field in the DP defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:
1. Program the DP_SELECT register’s APSEL field to choose one of the APs, and the APBANKSEL field to select the register bank to be accessed.
2. Program the A(3:2) field in the APACC register, if using JTAG, with the register address within the bank. Program the RnW bit to select a read or a write. In the case of a write, program the DATA field with the write data. If using SWD, the A(3:2) and RnW fields are part of the Packet Request word sent to the SW-DP with the APnDP bit set (see Table 763). The write data is sent in the data phase.

The debugger can access the memory mapped debug component registers through the MEM-AP registers (using the AP register access procedure described above) as follows:
1. Program the transaction target address in the TAR register.
2. Program the CSW register, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from the DRW register to initiate a bus transaction at the address held in the TAR register. Alternatively, a read or write to banked data register BDn triggers an access to address TAR[31:4] + n (this allows an access to the next four consecutive addresses without changing the address in the TAR register).

For more detailed information on the MEM-AP, refer to the Arm® Debug Interface Architecture Specification [1].

66.5.1 MEM-AP registers

<table>
<thead>
<tr>
<th>Address</th>
<th>APBANKSEL</th>
<th>A(3:2)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>0x0</td>
<td>0</td>
<td>AP_CSW</td>
<td>Control/status word register</td>
</tr>
<tr>
<td>0x04</td>
<td>0x0</td>
<td>1</td>
<td>AP_TAR</td>
<td>Transfer address register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Target address for the bus transaction.</td>
</tr>
<tr>
<td>0x08</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0C</td>
<td>0x0</td>
<td>3</td>
<td>AP_DRW</td>
<td>Data read/write register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:0]</td>
</tr>
<tr>
<td>0x10</td>
<td>0x1</td>
<td>0</td>
<td>AP_BD0</td>
<td>Banked data 0 register</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] &lt;&lt; 4 + 0x0</td>
</tr>
</tbody>
</table>
### Table 769. MEM-AP registers

<table>
<thead>
<tr>
<th>Address</th>
<th>APBANKSEL</th>
<th>A(3:2)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
| 0x14    | 0x1       | 1      | AP_BD1 | Banked Data 1 register  
Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x4 |
| 0x18    | 0x1       | 2      | AP_BD2 | Banked data 2 register  
Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0x8 |
| 0x1C    | 0x1       | 3      | AP_BD3 | Banked data 3 register  
Access to this register triggers a corresponding transaction on the debug bus to the address in TAR[31:4] << 4 + 0xC |
| 0x20-0xEC | -        | -      | -      | Reserved |
| 0xF0    | -         | -      | -      | Reserved |
| 0xF4    | -         | -      | -      | Reserved |
| 0x8     | 0xF       | 2      | AP_BASE | Debug base address register (RO)  
Base address of the ROM table |
| 0xFC    | 0xF       | 3      | AP_IDR | Identification register (RO) |
### Access port control/status word register (AP_CSW)

Address offset: 0x0

Reset value: 0x0000 0002 (APB-AP), 0x4000 0002 (AHB-AP)

| Bit 31 | Reserved, must be kept at reset value. |
| Bit 30 | SPROT: Secure transfer request bit |
|        | In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attribute HPROT[6] of the bus transfer. |
|        | 0: Reserved. |
|        | 1: Non-secure transfer. |
| Bit 29 | Reserved, must be kept at reset value. |

**Bits 28:24** PROT[4:0]: Bus transfer protection bits

In the APB-AP, this field is reserved. In the AHB-APs, this field sets the protection attributes HPROT[4:0] of the bus transfer.

- 0bXXX00: Instruction fetch
- 0bXXX01: Data access
- 0bXXX0X: User mode
- 0bXXX1X: Privileged mode
- 0bXX0XX: Non-bufferable
- 0bXX1XX: Bufferable
- 0bX00XX: Non-cacheable
- 0bX10XX: Cacheable
- 0b000XX: Non-exclusive
- 0b100XX: Exclusive

**Bit 23** SPISTATUS: Status of SPIDEN authentication signal

This bit determines whether the debugger can access secure memory. This field is reserved in the APB-AP.

- 0: Secure AHB transfers are not supported
- 1: Not used

**Bits 22:12** Reserved, must be kept at reset value.

**Bits 11:8** MODE[3:0]: Barrier support enabled bit

These bits define if the memory barrier operation is supported.

- 0x0: Not supported

**Bit 7** TRINPROG: Transfer in progress

This bit indicates that an AP bus transfer is in progress.

- 0: No transfer in progress.
- 1: Bus transfer in progress.
Bit 6 **DEVICEEN**: Device Enable bit
This bit defines whether the AP can be accessed or not.
1: AP access enabled.

Bits 5:4 **ADDRINC\[1:0\]**: Auto-increment mode bits
These bits define whether the TAR address is automatically incremented after a transaction.
0x0: no auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
0x2: Packed transfers enabled (Only in AHB-APs - reserved in APB-AP). A 32-bit AP access generates a 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transaction corresponding to the programmed transaction size. The data is packed or unpacked accordingly.
0x3: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE\[2:0\]**: Size of next memory access transaction (only for AHB-APs)
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
0x3-0x7: Reserved
For APB-AP, this field is read-only and fixed at 0x2 (32-bit).
**AP transfer address register (AP_TAR)**

Address offset: 0x04  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ADDRESS[31:16]</td>
</tr>
</tbody>
</table>

Bits 31:0 ADDRESS[31:0]: address of the current transfer (bits 31:0)

**AP data read/write register (AP_DRW)**

Address offset: 0x0C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DATA[31:16]</td>
</tr>
</tbody>
</table>

Bits 31:0 DATA[31:0]: read/write data for current transfer

In write mode, this is the write data value. In read mode, this is the read data value.

**AP banked data register 0 (AP_BD0)**

Address offset: 0x10  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>DATA[31:16]</td>
</tr>
</tbody>
</table>

Bits 31:0 DATA[31:0]: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0x0.
**AP banked data register 1 (AP_BD1)**

Address offset: 0x14

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**DATA[31:16]**

Bits 31:0  **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0x4.

**AP banked data register 2 (AP_BD2)**

Address offset: 0x18

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**DATA[15:0]**

Bits 31:0  **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0x8.

**AP banked data register 3 (AP_BD3)**

Address offset: 0x1C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**DATA[31:16]**

Bits 31:0  **DATA[31:0]**: read/write data for current transfer

The transaction address is TAR[31:4] << 4 + 0xC.
Access port base address register (AP_BASE)

Address offset: 0xF8
Reset value: 0xE00F E003 (AP1), 0xE00E 0003 (AP0)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12 BASEADDR[19:0]: Base address (bits 31 to 12) of the AP ROM table

The 12 LSBs are zero since the ROM table must be aligned on a 4 Kbyte boundary.

AP1 (Cortex-M7 AHB-AP): 0xE00FE
AP0 (System APB-AP): 0xE00E0

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 FORMAT: Base address register format
1: Arm® debug interface v5.

Bit 0 ENTRYPRESENT: Debug component present status bit
This bit indicates that debug components are present on the access port bus:
1: Debug components are present
## Access port identification register (AP_IDR)

Address offset: 0xFC

Reset value: 0x8477 0001 (AP1), 0x5477 0002 (AP0)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:28 **REVISION[3:0]**: Arm core revision
- 0x5: r1p0 (AP0)
- 0x8: r0p9 (AP1)

Bits 27:24 **JEDEC BANK[3:0]**: JEDEC bank
- 0x4: Arm®

Bits 23:17 **JEDEC CODE[6:0]**: JEDEC code
- 0x3B: Arm®

Bit 16 **MEMAP**: Memory access port
- 1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type identification
- 0x01: AHB-AP (AP1)
- 0x02: APB-AP (AP0)
66.5.2 Access port register map

These registers are not on the CPU memory bus, they are only accessed through SW-DP and JTAG-DP debug interfaces.


Table 770. Access port register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset value</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>AP_CSW</td>
<td>00000000</td>
<td>X 0 0 0 0 0 0 0</td>
<td>ADDRESS[31:0]</td>
</tr>
<tr>
<td>0x04</td>
<td>AP_TAR</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x08</td>
<td>Reserved</td>
<td>00000000</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0C</td>
<td>AP_DRW</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x10</td>
<td>AP_BD0</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x14</td>
<td>AP_BD1</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x18</td>
<td>AP_BD2</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x1C</td>
<td>AP_BD3</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x20</td>
<td>Reserved</td>
<td>00000000</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xF8</td>
<td>AP_BASE</td>
<td>00000000</td>
<td>1 1 1 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFC</td>
<td>AP_IDR</td>
<td>00000000</td>
<td>0 0 0 0 0 0 0 0</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
66.6 Trace and debug subsystem functional description

The trace and debug subsystem features the following CoreSight components:

- System ROM tables
- System cross-trigger interface (CTI)
- Cross-trigger matrix (CTM)
- Trace port interface unit (TPIU)
- Trace bus funnel (CSTF)
- Embedded trace FIFO (ETF)
- Serial wire output (SWO)

These components are accessible by the debugger via the system APB-AP and its associated APB-D debug bus. They are also accessible by the Cortex-M7 processor.

The MCU debug unit (DBGMCU) is also accessed via the APB-D. This component contains registers for configuring the device behavior in Debug mode.

A trace bus replicator branches the trace bus from the CPU’s ITM CoreSight component to ETF and SWO, through trace bus funnels.

66.6.1 System ROM tables

There are two ROM tables on the APB-D bus. The ROM table is a CoreSight component that contains the base addresses of all the CoreSight components on the APB-D bus. These tables allow a debugger to discover the topology of the CoreSight components automatically.

The first table points to the second table, and to the CoreSight components located in D3 power domain: SWO, DBGMCU. The table occupies a 4-Kbyte, 32-bit wide chunk of the APB-D address space, from 0xE00E0000 to 0xE00E0FFC when accessed by the debugger, and from 0x5C000000 to 0x5C000FFC when accessed from the system bus.
The second table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00F0000 to 0xE00F0FFC when accessed by the debugger, and from 0x5C010000 to 0x5C010FFC when accessed from the system bus.

The top of each ROM table contains a number of read-only registers, including the standard CoreSight component and peripheral identity registers, see section System ROM registers.

Each debug component occupies one or more 4 Kbyte blocks of address space. This block of address space is referred to as the debug register file for the component.
The component address offset field of a ROM Table entry points to the start of the last 4 Kbyte block of the address space of the component. This block always contains the component and peripheral ID registers for the component, starting at offset 0xFD0 from the start of the block. The 4 Kbyte count field PIDR4 [7:4], specifies the number of 4 Kbyte blocks for the component. Therefore, the process for finding the start of the address space for a component is:

1. Read the ROM-table entry for the component and extract its Address_Offset [18:0] from bits [31:12] of the ROM-table entry.
2. Use the address offset, together with the base address of the ROM table, ROM_Base_Address, to calculate the base address of the component:
   \[
   \text{Component\_Base\_Address} = \text{ROM\_Base\_Address} + \text{Address\_Offset}
   \]
   The Component_Base_Address is the start address of the 4 Kbyte block of the address space for the component.
3. Read the peripheral ID4 register for the component. The address of this register is:
   \[
   \text{Peripheral\_ID4\_address} = \text{Component\_Base\_Address} + 0xFD0
   \]
4. Extract the 4 Kbyte count field [7:4] from the value of the Peripheral ID4 Register.
5. Use the 4 Kbyte count field value to calculate the start address of the address space for the component. If the field value is 0b0000, which corresponds to a count value of 1, the address space for the component starts at Component_Base_Address obtained at stage 2.

The topology for the CoreSight components on the APB-D is shown in Figure 1005.
For more information on the use of the ROM table, refer to the Arm® Debug Interface Architecture Specification [1].
### 66.6.2 System ROM registers

**SYSROM memory type register (SYSROM_MEMTYPE)**

Address offset: 0xFCC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.  
Bit 0 **SYSMEM**: System memory  
0: No system memory is present on this bus

**SYSROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:4 **SIZE[3:0]**: Register file size  
0x0: Register file occupies a single 4 Kbyte region  
Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code  
0x0: STMicroelectronics JEDEC continuation code

**SYSROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)**

Address offset: 0xFE0  
Reset value: 0x0000 0085 (System ROM table 1), 0x0000 0001 (System ROM table 2)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>
**Debug infrastructure**

**SYSROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)**

Address offset: 0xFE4

Reset value: 0x0000 0004 (System ROM table 1), 0x0000 0000 (System ROM table 2)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PARTNUM[7:0]**: Device part number field, bits [7:0]

- 0x85: STM32H7Rx/7Sx device (System ROM table 1)
- 0x01: (System ROM table 2)

**SYSROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)**

Address offset: 0xFE8

Reset value: 0x0000 001A (system ROM table 1), 0x0000 000A (system ROM table 2)

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]

- 0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Device part number field, bits [11:8]

- 0x4: STM32H7Rx/7Sx (System ROM table 1)
- 0x0: (System ROM table 2)

**REVISION[3:0]**: Device revision number

- 0x1: (system ROM table 1)
- 0x0: (system ROM table 2)

Bit 3 **JEDEC**: JEDEC assigned value

- 1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]

- 0x2: STMicroelectronics JEDEC code
SYSROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)
Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0  **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SYSROM CoreSight component identity register 0 (SYSROM_CIDR0)
Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SYSROM CoreSight component identity register 1 (SYSROM_CIDR1)
Address offset: 0xFF4
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0D: Common ID value
SYSROM CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SYSROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SYSROM CoreSight component identity register 4 (SYSROM_CIDR4)

Address offset: 0xFFFF
Reset value: 0x0000 0001

Bits 31:8  Reserved, must be kept at reset value.
Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value
### 66.6.3 System ROM register map and reset values

#### Table 773. System ROM table 1 register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>SYSROM_MEMTYPE</td>
<td>0xFD0</td>
<td>SYSROM_PIDR4</td>
<td>0xFD4</td>
<td>Reserved</td>
<td>0xFE0</td>
<td>SYSROM_PIDR0</td>
<td>0xFE4</td>
<td>SYSROM_PIDR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xFE8</td>
<td></td>
<td>0xFE8</td>
<td></td>
<td>0xEFC</td>
<td></td>
<td>0xFF0</td>
<td></td>
<td>0xFF4</td>
</tr>
<tr>
<td></td>
<td>SYSROM_PIDR2</td>
<td></td>
<td>SYSROM_PIDR3</td>
<td></td>
<td>SYSROM_CIDR0</td>
<td></td>
<td>SYSROM_CIDR1</td>
<td></td>
<td>SYSROM_CIDR2</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td>0xFF8</td>
<td></td>
<td>0xFF8</td>
<td></td>
<td>0xFFC</td>
<td></td>
<td>0xFFC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYSROM_CIDR2</td>
<td></td>
<td>SYSROM_CIDR3</td>
<td></td>
<td>SYSROM_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xFFC</td>
<td></td>
<td>0xFFC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Reset values:

- 0xFCC: SYSROM_MEMTYPE
- 0xFD0: SYSROM_PIDR4
- 0xFE0: SYSROM_PIDR0
- 0xFE4: SYSROM_PIDR1
- 0xFE8: SYSROM_PIDR2
- 0xEFC: SYSROM_PIDR3
- 0xFF0: SYSROM_CIDR0
- 0xFF4: SYSROM_CIDR1
- 0xFF8: SYSROM_CIDR2
- 0xFFC: SYSROM_CIDR3
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>SYSROM_MEMTYPE</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_PIDR0</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>PARTNUM[7:0]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>0x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>SYSROM_PIDR4</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_PIDR1</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>PARTNUM[11:8]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>0x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_PIDR2</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>PARTNUM[11:8]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>0x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>SYSROM_PIDR3</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_CIDR0</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>PREAMBLE[7:0]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>0x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>SYSROM_CIDR1</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_CIDR1</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>CLASS[3:0]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>0x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF4</td>
<td>SYSROM_CIDR2</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_CIDR2</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>PREAMBLE[19:12]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x00</td>
<td>0x10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>SYSROM_CIDR3</td>
<td>0x00</td>
<td>0x10</td>
<td>SYSROM_CIDR3</td>
<td>0x00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>PREAMBLE[27:20]</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 774. System ROM table 2 register map and reset values
**66.6.4 Cross trigger interfaces (CTI) and matrix (CTM)**

The cross trigger interfaces (CTI) and cross trigger matrix (CTM) together form the CoreSight embedded cross trigger feature. There are two CTI components, one at system level and one dedicated to the Cortex-M7. The two CTIs are connected to each other via the CTM. The system-level CTI is accessible to the debugger via the system access port and associated APB-D. The Cortex-M7 CTI is physically integrated in the Cortex-M7 core, and is accessible via the Cortex-M7 access port and associated AHBD.

![Figure 1006. Embedded cross trigger](image)

The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a transition detected on an external trigger input can start code trace.

Each CTI has up to 8 trigger inputs and 8 trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in Table 775 to Table 778.
### Table 775. System CTI inputs

<table>
<thead>
<tr>
<th>#</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DBTRIGI</td>
<td>GPIO</td>
<td>External trigger input - allows an external signal to generate a debug event</td>
</tr>
<tr>
<td>1</td>
<td>ETFACQCOMP</td>
<td>ETF</td>
<td>ETF capture finished - allows a debug event to be generated when the trace FIFO is empty</td>
</tr>
<tr>
<td>2</td>
<td>ETFFULL</td>
<td>ETF</td>
<td>ETF full flag - allows a debug event to be generated when the trace FIFO is full</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>5</td>
<td>GPUFREEZE</td>
<td>GPU2D</td>
<td>GPU stopped</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Table 776. System CTI outputs

<table>
<thead>
<tr>
<th>#</th>
<th>Output signal</th>
<th>Destination component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DBTRIGO</td>
<td>GPIO</td>
<td>External IO trigger output - allows monitoring of events on the external DBTRIGO pin</td>
</tr>
<tr>
<td>1</td>
<td>TPIUFLUSH</td>
<td>TPIU</td>
<td>Trace port flush trigger - causes the TPIU FIFO to be flushed</td>
</tr>
<tr>
<td>2</td>
<td>TPIUTRIG</td>
<td>TPIU</td>
<td>Trace Port enable trigger - starts trace output on the external trace port</td>
</tr>
<tr>
<td>3</td>
<td>ETFTRIG</td>
<td>ETF</td>
<td>ETF enable trigger - starts filling the Trace FIFO</td>
</tr>
<tr>
<td>4</td>
<td>ETFFLUSH</td>
<td>ETF</td>
<td>ETF flush trigger - causes the Trace FIFO to be flushed</td>
</tr>
<tr>
<td>5</td>
<td>GPUFLAGSET</td>
<td>GPU2D</td>
<td>GPU freeze request</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>Not used</td>
</tr>
</tbody>
</table>

### Table 777. Cortex-M7 CTI inputs

<table>
<thead>
<tr>
<th>#</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HALTED</td>
<td>Cortex-M7 CPU</td>
<td>CPU halted - indicates CPU is in Debug mode</td>
</tr>
<tr>
<td>1</td>
<td>COMPMATCH0</td>
<td>Cortex-M7 DWT</td>
<td>DWT comparator 0 match</td>
</tr>
<tr>
<td>2</td>
<td>COMPMATCH1</td>
<td>Cortex-M7 DWT</td>
<td>DWT comparator 1 match</td>
</tr>
<tr>
<td>3</td>
<td>COMPMATCH2</td>
<td>Cortex-M7 DWT</td>
<td>DWT comparator 2 match</td>
</tr>
<tr>
<td>4</td>
<td>ETMEXTOUT0</td>
<td>Cortex-M7 ETM</td>
<td>ETM external trigger out</td>
</tr>
<tr>
<td>5</td>
<td>ETMEXTOUT1</td>
<td>Cortex-M7 ETM</td>
<td>ETM external trigger out</td>
</tr>
</tbody>
</table>
There are four event channels in the cross trigger matrix, which allows up to four parallel bidirectional connections between trigger inputs and outputs on different CTIs. To connect input number \( m \) on CTI \( x \) to output number \( n \) on CTI \( y \), the input must be connected to an event channel \( p \) using the \( \text{CTIINEN}_m \) register of CTI \( x \). The same channel \( p \) must be connected to the output using the \( \text{CTIOUTEN}_n \) register of CTI \( y \). Note: this applies even if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four), so an input can be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

**Figure 1007. Mapping of trigger inputs to outputs**

For more information on the cross-trigger interface CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2].

---

**Table 777. Cortex-M7 CTI inputs (continued)**

<table>
<thead>
<tr>
<th>#</th>
<th>Source signal</th>
<th>Source component</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
66.6.5 CTI registers

The register file base address for each CTI is defined by the ROM table for the bus to which it is connected. The registers are the same for each CTI.

**CTI control register (CTI_CONTROL)**

Address offset: 0x000
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 0 **GLBEN**: Global enable.
- 0: Cross-triggering disabled
- 1: Cross-triggering enabled

**CTI trigger acknowledge register (CTI_INTACK)**

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0 **INTACK[7:0]**: Trigger acknowledge

There is one bit of the register for each CTITRIGOUT output. When a 1 is written to a bit in this register, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.
CTI application trigger set register (CTI_APPSET)

Address offset: 0x014
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0  **APPSET[3:0]**: Set channel event

Read:
- 0bXXX0: Channel 0 event inactive
- 0bXXX1: Channel 0 event active
- 0bXX0X: Channel 1 event inactive
- 0bXX1X: Channel 1 event active
- 0bX0XX: Channel 2 event inactive
- 0bX1XX: Channel 2 event active
- 0b0XXX: Channel 3 event inactive
- 0b1XXX: Channel 3 event active

Write:
- 0bXXX0: No effect
- 0bXXX1: Set event on Channel 0
- 0bXX0X: No effect
- 0bXX1X: Set event on Channel 1
- 0bX0XX: No effect
- 0bX1XX: Set event on Channel 2
- 0b0XXX: No effect
- 0b1XXX: Set event on Channel 3

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**APPCLEAR[3:0]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**w**  w  w  w
CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  APPCLEAR[3:0]: Clear channel event
0bXXX0: No effect
0bXXX1: Clear event on Channel 0
0bX0X0: No effect
0bX1X: Clear event on Channel 1
0bX0XX: No effect
0bX1XX: Clear event on Channel 2
0b0XXX: No effect
0b1XXX: Clear event on Channel 3

CTI trigger IN x enable register (CTI_INENx)

Address offset: 0x0A0 + 0x4 * x, (x = 0 to 7)
Reset value: 0x0000 0000

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  APPPULSE[3:0]: Pulse channel event
This register clears itself immediately.
0bXXX0: No effect
0bXXX1: Generate pulse on Channel 0
0bX0X0: No effect
0bX1X: Generate pulse on Channel 1
0bX0XX: No effect
0bX1XX: Generate pulse on Channel 2
0b0XXX: No effect
0b1XXX: Generate pulse on Channel 3
Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **TRIGINEN[3:0]**: Cross-trigger event enable

Enables or disables a cross-trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).

- 0bXXX0: Trigger n does not generate events on Channel 0
- 0bXXX1: Trigger n generates events on Channel 0
- 0bXXOX: Trigger n does not generate events on Channel 1
- 0bXX1X: Trigger n generates events on Channel 1
- 0bX0XX: Trigger n does not generate events on Channel 2
- 0bX1XX: Trigger n generates events on Channel 2
- 0b0XXX: Trigger n does not generate events on Channel 3
- 0b1XXX: Trigger n generates events on Channel 3

**CTI trigger OUT x enable register (CTI_OUTENx)**

Address offset: 0x0A0 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

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

Bits 3:0  **TRIGOUTEN[3:0]**: Enable trigger upon event

For each channel, the field defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

- 0bXXX0: Channel 0 events do not generate triggers on Trigger output n
- 0bXXX1: Channel 0 events generate triggers on Trigger output n
- 0bXX0X: Channel 1 events do not generate triggers on Trigger output n
- 0bXX1X: Channel 1 events generate triggers on Trigger output n
- 0bX0XX: Channel 2 events do not generate triggers on Trigger output n
- 0bX1XX: Channel 2 events generate triggers on Trigger output n
- 0b0XXX: Channel 3 events do not generate triggers on Trigger output n
- 0b1XXX: Channel 3 events generate triggers on Trigger output n

**CTI trigger IN status register (CTI_TRGISTS)**

Address offset: 0x130

Reset value: 0x0000 0000

Links to official STMicroelectronics documentation for further details.
Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: Trigger input status
There is one bit of the register for each CTITRIGIN input. When a bit is set to 1 it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

**CTI trigger OUT status register (CTI_TRGOSTS)**
Address offset: 0x134
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status
There is one bit of the register for each CTITRIGOUT output. When a bit is set to 1 it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

**CTI channel IN status register (CTI_CHINSTS)**
Address offset: 0x138
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0 **CHINSTATUS[3:0]**: Channel input status
There is one bit of the register for each channel input. When a bit is set to 1 it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.
**CTI channel OUT status register (CTI_CHOUTSTS)**

Address offset: 0x13C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

Bits 3:0  **CHOUTSTATUS[3:0]**: Channel output status  
There is one bit of the register for each channel output. When a bit is set to 1 it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

**CTI channel gate register (CTI_GATE)**

Address offset: 0x140  
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

Bits 3:0  **GATEEN[3:0]**: Channel output enable  
For each channel, defines whether an event on that channel can propagate over the CTM to other CTIs.  
0bXXX0: Channel 0 events do not propagate  
0bXXX1: Channel 0 events propagate  
0bXX0X: Channel 1 events do not propagate  
0bXX1X: Channel 1 events propagate  
0bX0XX: Channel 2 events do not propagate  
0bX1XX: Channel 2 events propagate  
0b0XXX: Channel 3 events do not propagate  
0b1XXX: Channel 3 events propagate
CTI claim tag set register (CTI_CLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

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

Bits 3:0  CLAIMSET[3:0]: Set claim tag bits
Write:
0000: No effect
0001: Set bit 0
xx1x: Set bit 1
xx1x: Set bit 2
1xxx: Set bit 3
Read:
0xF: Indicates there are four bits in claim tag

CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

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

Bits 3:0  CLAIMCLR[3:0]: Reset claim tag bits
Write:
0000: No effect
0001: Clear bit 0
xx1x: Clear bit 1
xx1x: Clear bit 2
1xxx: Clear bit 3
Read: Returns current value of claim tag
**CTI lock access register (CTI_LAR)**

Address offset: 0xFB0  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0 **ACCESS_W[31:0]**: CTI register write access enable  
* Enables write access to some CTI registers by processor core (debuggers do not need to unlock the component)  
* 0xC5ACCE55: Enable write access  
* Other values: Disable write access

**CTI lock status register (CTI_LSR)**

Address offset: 0xFB4  
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 2 **LOCKTYPE**: Size of the CTI_LAR register  
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock  
This bit always returns zero when read by an external debugger.  
0: Write access is permitted  
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism  
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.  
0: No lock control mechanism exists  
1: Lock control mechanism is implemented
CTI authentication status register (CTI_AUTHSTAT)

Address offset: 0xFB8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SNID[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Sid[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NSNID[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NSID[1:0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **Sid[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

CTI device configuration register (CTI_DEVID)

Address offset: 0xFC8
Reset value: 0x0004 0800

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available
0x4: 4 channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available
0x8: 8 trigger inputs and 8 trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None
CTI device type identifier register (CTI_DEVTYPE)

Address offset: 0xFCC
Reset value: 0x0000 0014

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

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

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: Indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

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

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0 **JEPO6CON[3:0]**: JEPO6 continuation code
0x4: Arm® JEDEC code
### CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

**Address offset:** 0xFE0  
**Reset value:** 0x0000 0006

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 7:0:** **PARTNUM[7:0]:** Part number field, bits [7:0]  
0x06: CTI part number

### CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

**Address offset:** 0xFE4  
**Reset value:** 0x0000 00B9

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

**Bits 3:0:** **PARTNUM[11:8]:** Part number field, bits [11:8]  
0x9: CTI part number

### CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

**Address offset:** 0xFE8  
**Reset value:** 0x0000 005B

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bits 7:4:** **REVISION[3:0]:** Revision field, bits [3:0]  
0: 0x04

**Bits 3:0:** **JEDEC:** JEDEC standard

**Bits 2:0:** **JEP106ID[6:4]:** JEP106 identity code field, bits [6:4]  
0x7: JEDEC code
CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

Bits 7:4 **REVISION[3:0]**: Component revision number
0x5: r1p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value
### CTI CoreSight component identity register 1 (CTI_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class  
0x9: CoreSight component  
Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]  
0x0: Common ID value

### CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8  
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |
|      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |      |

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]  
0x05: Common ID value

### CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC  
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]

0xB1: Common ID value
### 66.6.6 CTI register map and reset values

**Table 779. CTI register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Address Range</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>CTI_CONTROL</td>
<td>31-0</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x010</td>
<td>CTI_INTACK</td>
<td>INTACK[7:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x014</td>
<td>CTI_APPSET</td>
<td>APPSET[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x018</td>
<td>CTI_APPCLEAR</td>
<td>APPCLEAR[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x01C</td>
<td>CTI_APPPULSE</td>
<td>APPPULSE[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x020 to</td>
<td>CTI_INEN0 to</td>
<td>TRIGINEN[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x03C</td>
<td>CTI_OUTEN0 to</td>
<td>TRIGOUTEN[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x040 to</td>
<td>Reserved</td>
<td></td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x09C</td>
<td>CTI_OUTEN0 to</td>
<td>TRIGOUTEN[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x0C0 to</td>
<td>Reserved</td>
<td></td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x130</td>
<td>CTI_TRIGISTS</td>
<td>TRIGINSTATUS[7:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x134</td>
<td>CTI_TRIGOSTS</td>
<td>TRIGOUTSTATUS[7:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x138</td>
<td>CTI_CHINSTS</td>
<td>CHSTATUS[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x13C</td>
<td>CTI_CHOUTSTS</td>
<td>CHOSTATUS[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
<tr>
<td>0x140</td>
<td>CTI_GATE</td>
<td>GATEEN[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 1</td>
</tr>
<tr>
<td>0x144 to</td>
<td>Reserved</td>
<td></td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 1</td>
</tr>
<tr>
<td>0xF0</td>
<td>CTICLAIMSET</td>
<td>CLAIMSET[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 1</td>
</tr>
<tr>
<td>0xFA4</td>
<td>CTICLAIMCLR</td>
<td>CLAIMCLR[3:0]</td>
<td>Reset value: - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 0</td>
</tr>
</tbody>
</table>

Legend:
- **GUBEN**: General Use Bit Enable
- **INTACK**: Interrupt Acknowledge
- **APPSET**: Application Set
- **APPCLEAR**: Application Clear
- **APPPULSE**: Application Pulse
- **TRIGINEN**: Trigger Input Enable
- **TRIGOUTEN**: Trigger Output Enable
- **CHSTATUS**: Channel Status
- **CHOSTATUS**: Channel Output Status
- **GATEEN**: Gate Enable
- **CLAIMSET**: Claim Set
- **CLAIMCLR**: Claim Clear
### RM0477 Debug infrastructure

#### Table 779. CTI register map and reset values (continued)

| Offset   | Register name     | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|----------|------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFB0    | CTI_LAR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB4    | CTI_LSR          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB8    | CTI_AUTHSTAT     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | Reset value      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC8    | CTI_DEVID        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | NUMCH[3:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | NUMTRIG[7:0]     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF0    | CTI_CIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | PREAMBLE[7:0]    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFF4    | CTI_CIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | CLASS[3:0]       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | PREAMBLE[11:8]   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFFF    | CTI_CIDR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | PREAMBLE[19:12]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFFC    | CTI_CIDR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|          | PREAMBLE[27:20]  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
66.6.7 Trace funnel (CSTF)

The trace funnel is a CoreSight component that combines the ATB buses from two trace sources into one single ATB. The CSTF has two ATB slave ports, and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:
- S0: Cortex-M7 ETM
- S1: Cortex-M7 ITM

The CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when the trace is disabled. The arbitration works as follows:
- The arbiter selects the slave port with the highest assigned priority that has data valid
- Transfers are passed from the selected slave to the master port until *min_hold_time is reached*, where *min_hold_time* is programmable in the CONTROL register.
- A new arbitration is then performed

High priority should be assigned to slave ports connected to sources with a small amount of buffering, or where data loss can not be tolerated. Low priority should be assigned to less critical sources or those with large buffers.

For more information on the ATB Funnel CoreSight component, refer to the Arm® CoreSight™ SoC-400 Technical Reference Manual [2].
### 66.6.8 Trace funnel registers

**CSTF control register (CSTF_CTRL)**

Address offset: 0x000  
Reset value: 0x0000 0300

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>224</td>
</tr>
<tr>
<td>Bits 31:12 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 11:8** `MIN_HOLD_TIME[3:0]`: Number of transactions between arbitrations.  
- 0x0: 1 transaction  
- 0xE: 15 transactions  
- 0xF: Reserved

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

**Bit 1** `ENS1`: S1 slave port enable  
0: Disable port  
1: Enable port

**Bit 0** `ENS0`: S0 slave port enable  
0: Disable port  
1: Enable port

**CSTF priority register (CSTF_PRIORITY)**

Address offset: 0x004  
Reset value: 0x0000 0688

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>224</td>
</tr>
<tr>
<td>Bits 31:12 Reserved, must be kept at reset value.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Bits 15:8** `PRIPORT1[2:0]`:  

**Bits 7:0** `PRIPORT0[2:0]`:  

---

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

Bits 5:3  **PRIPORT1[2:0]**: S1 slave port priority
  0: Highest priority
  7: Lowest priority

Bits 2:0  **PRIPORT0[2:0]**: S0 slave port priority
  0: Highest priority
  7: Lowest priority

**CSTF claim tag set register (CSTF_CLAIMSET)**

Address offset: 0xFA0

Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 3:0  **CLAIMSET[3:0]**: Set claim tag bits

Write:
  0000: No effect
  xxx1: Set bit 0
  xx1x: Set bit 1
  x1xx: Set bit 2
  1xxx: Set bit 3

Read:
  0xF: Indicates there are four bits in claim tag

**CSTF claim tag clear register (CSTF_CLAIMCLR)**

Address offset: 0xFA4

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

3618/3791  RM0477 Rev 6
Bits 31:4  Reserved, must be kept at reset value.

Bits 3:0  **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3
Read: Returns current value of claim tag

**CSTF lock access register (CSTF_LAR)**
Address offset: 0xFB0
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**Access W[31:16]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**Access W[15:0]**

Bits 31:0  **ACCESS_W[31:0]**: CSTF register write access enable
The field enables write access to some CSTF registers by the processor cores (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access
Other values: Disable write access

**CSTF lock status register (CSTF_LSR)**
Address offset: 0xFB4
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

**Access W[63:16]**

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:3  Reserved, must be kept at reset value.
Bit 2 **LOCKTYPE**: Size of the CSTF_LAR register
- 0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
- 0: Write access is permitted
- 1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
- 0: No lock control mechanism exists
- 1: Lock control mechanism is implemented

**CSTF authentication status register (CSTF_AUTHSTAT)**

Address offset: 0xFB8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
- 0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
- 0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
- 0x2: Disabled
- 0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
- 0x2: Disabled
- 0x3: Enabled

**CSTF CoreSight device identity register (CSTF_DEVID)**

Address offset: 0xFC8
Reset value: 0x0000 0024

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3620/3791
### CSTF CoreSight device type identity register (CSTF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0012

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td><strong>SCHEME[3:0]</strong>: Priority scheme</td>
</tr>
<tr>
<td>0x2: Static priority</td>
<td></td>
</tr>
<tr>
<td>Bits 3:0</td>
<td><strong>PORTCNT[3:0]</strong>: Number of input ports connected</td>
</tr>
<tr>
<td>0x4: Four input ports</td>
<td></td>
</tr>
</tbody>
</table>

### CSTF CoreSight peripheral identity register 4 (CSTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:0</td>
<td><strong>DEVTYPEID[7:0]</strong>: Device type identifier</td>
</tr>
<tr>
<td>0x12: Trace funnel</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 31:8</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bits 7:4</td>
<td><strong>SIZE[3:0]</strong>: Register file size</td>
</tr>
<tr>
<td>0x0: Register file occupies a single 4 Kbyte region</td>
<td></td>
</tr>
<tr>
<td>Bits 3:0</td>
<td><strong>JEP106CON[3:0]</strong>: JEP106 continuation code</td>
</tr>
<tr>
<td>0x4: Arm® JEDEC code</td>
<td></td>
</tr>
</tbody>
</table>
### CSTF CoreSight peripheral identity register 0 (CSTF_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]  
0x08: CSTF part number

### CSTF CoreSight peripheral identity register 1 (CSTF_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

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

Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]  
0x9: CSTF part number

### CSTF CoreSight peripheral identity register 2 (CSTF_PIDR2)

Address offset: 0xFE8  
Reset value: 0x0000 003B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]**: Revision code field, bits [3:0]

Bits 3:0  **JEDEC JEP106ID[6:4]**: JEDEC code field, bits [6:4]
Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x3: r1p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code

**CSTF CoreSight peripheral identity register 3 (CSTF_PIDR3)**
Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

**CSTF CoreSight component identity register 0 (CSTF_CIDR0)**
Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value
CSTF CoreSight component identity register 1 (CSTF_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  CLASS[3:0]: Component ID field, bits [15:12] - component class
0x9: CoreSight component

0x0: Common ID value

CSTF CoreSight component identity register 2 (CSTF_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  PREAMBLE[19:12]: Component ID field, bits [23:16]
0x05: Common ID value

CSTF CoreSight component identity register 3 (CSTF_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  PREAMBLE[27:20]: Component ID field, bits [23:16]
0x05: Common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value
## 66.6.9 Trace funnel register map and reset values

### Table 780. CSTF register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>CSTF_CTRL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000001111 - 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>CSTF_PRIORITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000001010 - 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x008  to 0x0F9C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FA</td>
<td>CSTF_CLAIMSET</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000001111 - 11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FA4</td>
<td>CSTF_CLAIMCLR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000001111 - 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FB0</td>
<td>CSTF_LAR</td>
<td>ACCESS_W[31:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000000000 - 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FB4</td>
<td>CSTF_LSR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000000000 - 01</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FB8</td>
<td>CSTF_AUTHSTAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000000101 - 00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FC8</td>
<td>CSTF_DEVID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>00000000000000 - 00</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>----------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>0xFCC</td>
<td>CSTF_DEVTYPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 0 1 0 0 1 0</td>
</tr>
<tr>
<td>0xFD0</td>
<td>CSTF_PIDR4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0xFD4 to 0xFD8 Reserved Reserved</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>CSTF_PIDR0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 0 0 1 1 0 1</td>
</tr>
<tr>
<td>0xFE4</td>
<td>CSTF_PIDR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>1 0 1 1 1 0 0 1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>CSTF_PIDR2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 1 1 1 0 1 1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>CSTF_PIDR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0xFF0</td>
<td>CSTF_CIDR0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 0 1 0 1 0 1</td>
</tr>
<tr>
<td>0xFF4</td>
<td>CSTF_CIDR1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>1 0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>0xFF8</td>
<td>CSTF_CIDR2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>0 0 0 0 0 1 0 1</td>
</tr>
<tr>
<td>0xFFC</td>
<td>CSTF_CIDR3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - -</td>
<td>1 0 1 1 0 0 0 1</td>
</tr>
</tbody>
</table>
66.6.10 Embedded trace FIFO (ETF)

The ETF is a 2 Kbyte memory that captures trace data from two trace sources, namely the ETM and ITM of the CPU core. The ETF is a design configuration of the CoreSight™ trace memory controller component.

The ETF can be used in three modes (selected in the mode register):

1. Hardware FIFO mode
   - The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured by the trace RAM and when full, the incoming trace stream is stalled. When the Trace buffer is not empty, trace data is drained out through the ATB master interface to the TPIU.
   - In this mode, the role of the FIFO is to smooth the flow of trace information arriving at the trace port. Since the trace data can be very bursty by nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, which can then be sized according to the average rate rather than the peak. The trace is stored off-chip in real time by the trace port analyzer tool, and so the trace log can be very big.

2. Software FIFO mode
   - The trace memory is used as a FIFO that can be read through the RRD Register while the trace is being captured. Trace data is captured by the trace RAM and when full, the incoming trace stream is stalled.
   - This mode allows the trace to be transferred by DMA into the system memory, or to a high speed interface (SPI, USB and so on), or even monitored by software. Note that unlike the hardware FIFO mode, this mode is invasive, since it uses system resources which are shared by the processor.

3. Circular buffer mode
   - The trace memory is used as a circular buffer. Trace data is captured in the trace memory starting from the location pointed to by the write pointer register. When the trace memory is full, incoming trace data continues to be overwritten in the trace memory until a stop condition occurs.
   - In this mode, the ETF stores the trace data on-chip, so the trace log size is limited to that of the ETF SRAM, 2 Kbytes in this case. Being a circular buffer, when the FIFO is full, incoming trace data overwrites the oldest stored data and the oldest stored data is lost. Therefore the content of the trace buffer represents the most recent activity of the processor, up to the point when the buffer was stopped, rather than all the activity since the trace was started.

   There are three possible methods to read the buffer contents once the trace stops:
   - via the Trace port - with the TPIU enabled, the contents of the buffer are output over the Trace port. This can be done by setting the DRAINBUF bit in the ETF_FFCR register.
   - via the Debug port - the debugger can read the buffer via the RRD register that is accessible over the system APB-D.
   - by software - the processor can read the buffer via the RRD register, since the APB-D is accessible from the system bus.
The ETF can be moved to any one of these states:

- **Disabled**
  This state is entered after a reset, or when trace capture is disabled. The ETF must only be programmed in this state.

- **Running**
  Trace capture is performed in this state. It is entered by enabling trace capture while in Disabled state.

- **Stopped**
  Trace capture is stopped in this state, but the contents of the buffer can be read out or drained. This state is entered after a stop event (trigger or flush).

- **Disabling**
  This is a transition state while disabling trace capture.

- **Stopping**
  This is a transition state while stopping trace capture.

- **Draining**
  This state is entered while draining the buffer in Stopped state.

The state transition diagram is shown in Figure 1008.

**Figure 1008. ETF state transition diagram**

For more information on the CoreSight™ trace memory controller component, refer to the Arm® CoreSight™ trace memory controller technical reference manual [3].
66.6.11 ETF registers

ETF RAM size register (ETF_RSZ)

Address offset: 0x004
Reset value: 0x0000 0200

<table>
<thead>
<tr>
<th>Bit</th>
<th>RSZ[30:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
</tr>
</tbody>
</table>

Bit 31  Reserved, must be kept at reset value.

Bits 30:0  **RSZ[30:0]**  RAM size

The value of the field indicates the number of 32-bit words
0x200: 512 words = 2 Kbytes

ETF status register (ETF_STS)

Address offset: 0x00C
Reset value: 0x0000 001C

<table>
<thead>
<tr>
<th>Bit</th>
<th>EMTY</th>
<th>FTEMP</th>
<th>READY</th>
<th>TRIGD</th>
<th>FULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>30</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>29</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>28</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>27</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>26</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>25</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>24</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>23</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>22</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>21</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>20</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>18</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>17</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>16</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bit 4  **EMPTY**: Trace FIFO empty

This bit is valid only when the TCEN bit of the ETF_CTL register is high. This bit reads as zero when TCEN is low.
0: Trace FIFO contains data
1: Trace FIFO is empty.

Note: Empty trace FIFO does not mean that the ETF pipeline is empty. The latter is indicated by the FTEMPY bit.

Bit 3  **FTEMPY**: Formatter empty

This bit is set when a trace capture has stopped, and all internal pipelines and buffers have been drained. Unlike READY, it is not affected by buffer drains. The ACQCOMP output reflects the value of this bit.
Bit 2 **READY**: ETF ready
   This bit is set when a trace capture has stopped and all internal pipelines and buffers have been drained (Stopped or Disabled state)

Bit 1 **TRIGD**: Triggered
   The Triggered bit is set when a trace capture is in progress and the TMC has detected a Trigger Event. This bit is cleared when leaving Disabled state.
   This bit is operational only in the Circular buffer mode. In all other modes, this bit is always low.
   This bit does not indicate that a trigger has been embedded in the formatted output trace data from the TMC. Trigger indication on the output trace stream is determined by the programming of the Formatter and Flush Control Register, ETF_FFCR.

Bit 0 **FULL**: Trace buffer full
   In circular buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until the TCEN bit of the ETF_CTL register is cleared and set.
   In software and hardware FIFO modes, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in the ETF_BUFWM Register, that is, Fill level >= MEM_SIZE - BUFWM.
   This bit is cleared when leaving Disabled state. The FULL output reflects the value of this register bit.

**ETF RAM read data register (ETF_RRD)**

Address offset: 0x010
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15:0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **RRD[31:0]**: RAM Read Data.
   **Circular buffer mode:**
   When in Stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer. When the whole trace buffer has been read, the Empty bit in the ETF_STS Register is set, and subsequent reads return 0xFFFFFFF. Reading this register when not in Stopped state returns 0xFFFFFFFF.
   **Software FIFO mode:**
   Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, the data returned is 0xFFFFFFFF.
   **Hardware FIFO mode:**
   Reading this register returns 0xFFFFFFFF.
ETF RAM read pointer register (ETF_RRP)

Address offset: 0x014
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **RRP[12:0]: RAM Read Pointer**

The RAM Read Pointer Register contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via the ETF_RRD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been written. When the pointer reaches its maximum value, it wraps around.

This register can only be written to while in Disabled state. It can be read in Disabled state, in Stopped state in Circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.

ETF RAM write pointer register (ETF_RWP)

Address offset: 0x018
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **RWP[12:0]: RAM write pointer**

The RAM write pointer register contains the value of the write pointer that is used to write entries into the trace memory over the APB interface via the ETF_RWD register. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 3 should be zero). The pointer is incremented by 8 each time a full 64-bit FIFO entry has been read. When the pointer reaches its maximum value, it wraps around.

This register can only be written to while in Disabled state. It can be read in Disabled state, in Stopped state in Circular buffer mode and SW FIFO mode, and also in Running and Stopping states in SW FIFO mode.
ETF trigger counter register (ETF_TRG)

Address offset: 0x01C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11  Reserved, must be kept at reset value.

Bits 10:0  TRG[10:0]: Trigger counter

In Circular buffer mode, specifies the number of 32-bit words to capture in the trace RAM following the detection of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream, ATID = 7’h7D. On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETF behavior is controlled by the FFCR Register.

The number of 32-bit words written into the trace RAM following the trigger is the value stored in this register, plus one. This register is ignored when the ETF is in Software FIFO mode or Hardware FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches zero. When the trigger counter has reached zero, it remains at zero until it is re-programmed with a write to this register.

This register is cleared when READY goes high, so that the state of the counter when trace capture has stopped does not affect a subsequent trace capture session. Writing to this register when not in Disabled state results in unpredictable behavior.

A read access to this register is permitted at any time when in Disabled state, or in Circular buffer mode. A read access returns the current value of the trigger counter.

ETF control register (ETF_CTL)

Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 0  TCEN: Trace capture enable

When writing:
0: Disable trace capture (moves from Running, Stopping or Stopped state into Disabling or Disabled state)
1: Enable trace capture (moves from Disabled state to Running state)

When reading, this bit is low when in Disabling or Disabled states, and high otherwise.
ETF RAM write data register (ETF_RWD)

Address offset: 0x024
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th></th>
<th>RWD[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>RWD[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  **RWD[31:0]**: RAM write data
When in Disabled state, a write to this register stores data at the location pointed to by the RWP. Writes to this register when not in Disabled state are ignored. When a full memory width (64-bit) of data has been written, the data is written to memory and the RAM Write Pointer is incremented to the next memory word.
This register is used for test purposes.

ETF mode register (ETF_MODE)

Address offset: 0x028
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>MODE[1:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 1:0  **MODE[1:0]**: Operation mode

00b: Circular buffer mode
In this mode, the trace memory is used as a circular buffer. Trace data is captured in the trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continues to be overwritten into the trace memory until a stop condition has occurred.

01b: Software FIFO mode
In this mode, the trace memory is used as a FIFO that can be read through the RRD Register while a trace is being captured. Trace data is captured in the trace RAM and when full, the incoming trace stream is stalled.

10b: Hardware FIFO mode
In this mode, the trace memory is used as a FIFO that is drained through the ATB master interface. Trace data is captured in the trace RAM and when full, the incoming trace stream is stalled. When the trace buffer is non-empty, trace data is drained out through the ATB master interface to the TPIU.

11b: Reserved
ETF latched buffer fill level register (ETF_LBUFLVL)

Address offset: 0x02C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  LBUFLEVEL[11:0]: Latched buffer fill level

- Reading this register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.
- When entering Disabled state, this register retains its last value. While in Disabled state, reads from this register do not affect its value. When exiting Disabled state, the LBUFLEVEL register is cleared.
- This register is used for performance analysis of the trace system.

ETF current buffer fill level register (ETF_CBUFLVL)

Address offset: 0x030
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  CBUFLEVEL[11:0]: Current buffer fill level

- Reading this register returns the current fill level of the trace memory in 32-bit words.
- This register is cleared when TCEN is low.
ETF buffer level watermark register (ETF_BUFWM)

Address offset: 0x034
Reset value: 0x0000 0000

| Bits 31:11 | Reserved, must be kept at reset value. |
| Bits 10:0 | BUFWM[10:0]: Buffer level watermark |

The value programmed into this register indicates the required threshold vacancy level of the trace memory in 32-bit words. When the space in the FIFO is less than or equal to this value, that is, fill level >= MEM_SIZE - BUFWM, the FULL output is pulled high and the FULL bit in the STS Register is set.

This register is used only in Software FIFO and Hardware FIFO modes. In Circular buffer mode, this functionality can be obtained by programming the RWP to the required vacancy trigger level, so that when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.

The maximum value that can be written into this register is MEM_SIZE - 1. In this case, the FULL bit output is asserted after the first 32-bit word is written to trace memory.

Writing to this register other than when in disabled state results in unpredictable behavior.

ETF formatter and flush status register (ETF_FFSR)

Address offset: 0x300
Reset value: 0x0000 0002

| Bits 31:2 | Reserved, must be kept at reset value. |
| Bit 1 | FTSTOPPED: Formatter stopped |

This bit behaves in the same way as the FEMPTY bit in the ETF_STS register.

| Bit 0 | FLINPROG: Flush in progress |

Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the ETF_FFCR register, or requested by the ATB master port.

0: No flush in progress
1: Flush in progress
ETF formatter and flush control register (ETF_FFCR)

Address offset: 0x304
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>30</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>29</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>28</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>26</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>25</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>24</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>22</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>21</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>20</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>19</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>18</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>17</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>16</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>15</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>DRAINBUF: Drain buffer</td>
</tr>
<tr>
<td>13</td>
<td>STPONTRGEV: Stop on trigger event</td>
</tr>
<tr>
<td>12</td>
<td>STOPONFL: Stop on flush</td>
</tr>
<tr>
<td>11</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>10</td>
<td>TRIGONFL: Trigger on flush</td>
</tr>
</tbody>
</table>

Bits 31:15  Reserved, must be kept at reset value.

Bit 14  DRAINBUF: Drain buffer
This bit is used to enable draining of the trace data through the ATB master interface after the formatter has stopped. This is useful in circular buffer mode to capture trace data into trace memory and then to drain the captured trace through the ATB master interface.
Writing a 1 to this bit when in Stopped state starts the drain of the trace buffer contents through the ATB Master interface. This bit always reads as zero. The READY bit in the ETF_STS register goes low while the drain is in progress.
This bit is only functional when the ETF is in Circular buffer mode and formatting is enabled, that is, the ENFT bit in the ETF_FFCR register is set. Setting this bit when the ETF is in any other mode, or when not in Stopped state, results in Unpredictable behavior.
When trace capture is complete in Circular buffer mode, all of the captured trace must be retrieved from the trace memory through the same mechanism, either read all trace data out through RRD reads, or drain all trace data by setting the DRAINBUF bit. Setting the DRAINBUF bit after some of the captured trace has been read out through RRD results in unpredictable behavior.

Bit 13  STPONTRGEV: Stop on trigger event
0: No effect
1: Stop trace capture when a trigger event occurs
Enabling the ETF in Software FIFO mode or Hardware FIFO mode with this bit set results in unpredictable behavior.

Bit 12  STOPONFL: Stop on flush
0: No effect
1: Stop trace capture when flush is completed
If a flush is initiated by the ATB master interface, its completion does not lead to a formatter stop regardless of the value programmed in this bit.

Bit 11  Reserved, must be kept at reset value.

Bit 10  TRIGONFL: Trigger on flush
0: No effect
1: Indicates a trigger in the trace stream when flush is completed
If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.
If a flush is initiated by the ATB master interface, its completion does not lead to a trigger indication regardless of the value programmed in this bit.
Bit 9 **TRGONTRGEV**: Trigger on trigger event
   0: No effect
   1: Indicates a trigger in the trace stream when a trigger event occurs
   If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.
   This bit is not supported in Software FIFO mode nor Hardware FIFO mode.

Bit 8 **TRGONTRGIN**: Trigger on trigger in
   0: No effect
   1: Indicate a trigger in the trace stream when a rising edge is detected on the TRIGIN input.
   If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FLUSHMAN**: Manual flush
   0: No effect
   1: Flush the trace FIFO and pipeline
   This bit is cleared automatically when the flush is complete. If the TCEN bit in the ETF_CTL register is 0, writes to this bit are ignored.

Bit 5 **FONTRGEV**: Flush on trigger event
   0: No effect
   1: Flush the trace FIFO and pipeline if a trigger event occurs
   This bit is not supported in Software FIFO mode or Hardware FIFO mode. If STPONTRGEV is set, this bit is ignored.

Bit 4 **FONFLIN**: Flush on flush in
   0: No effect
   1: Flush the trace FIFO and pipeline if when a rising edge is detected on the FLUSHIN input

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENTI**: Enable trigger insertion
   Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 8'h00 with ATID 7'h7D in the trace stream. Trigger indication on the trace stream is further controlled by the register bits TRIGONFL, TRGONTRGEV, and TRGONTRGIN in the FFCR Register. This bit can only be changed when READY is high, and TCEN is low. This bit takes effect only when the ENFT register bit in this register is set. If ENTI bit is set to high when ENFT is low, it results in formatting being enabled.

Bit 0 **ENFT**: Enable formatting.
   0: Formatting is disabled. Incoming trace data is assumed to be from a single trace source.
   1: Formatting is enabled.
   If multiple ATIDs are received by the ETF when trace capture is enabled and the formatter is disabled, an interleaving of trace data occurs. Disabling of formatting is supported only in Circular buffer mode. If the ETF is enabled in a mode other than Circular buffer mode with ENFT low, then formatting is enabled. If ENTI bit is set to high when ENFT is low, formatting is enabled.
   This bit is ignored when in Disabled state.
ETF periodic synchronization counter register (ETF_PSCR)

Address offset: 0x308
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 4:0 \texttt{PSCOUNT}[4:0]: Synchronization counter reload value

Determines the reload value of the Synchronization Counter. The reload value takes effect the next time the counter reaches zero. Reads from this register return the reload value programmed into this register. This register is set to 0xA on reset, corresponding to a synchronization period of 1024 bytes.

- 0x0: Synchronization disabled
- 0x1-0x6: Reserved
- 0x7-0x1B: Synchronization period is \(2^\text{PSCOUNT}\) bytes
- 0x1C-0x1F: Reserved

ETF claim tag set register (ETF_CLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 3:0 \texttt{CLAIMSET}[3:0]: Set claim tag bits

Write:
- 0000: No effect
- xxx1: Set bit 0
- xx1x: Set bit 1
- x1xx: Set bit 2
- 1xxx: Set bit 3

Read:
- 0xF: Indicates there are four bits in claim tag
ETF claim tag clear register (ETF_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0  CLAIMCLR[3:0]: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3

Read: Returns current value of claim tag

ETF lock access register (ETF_LAR)

Address offset: 0xFB0
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:0  ACCESS_W[31:0]: ETF register access enable

Enables write access to some ETF registers by the processor cores (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access
Other values: Disable write access
ETF lock status register (ETF_LSR)

Address offset: 0xFB4
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 2 **LOCKTYPE**: Size of the ETF_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns zero when read by an external debugger.
0: Write access is permitted
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns zero when read
by an external debugger.
0: No lock control mechanism exists
1: Lock control mechanism is implemented

ETF authentication status register (ETF_AUTHSTAT)

Address offset: 0xFB8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Res</td>
<td>Res</td>
<td>SNID[1:0]</td>
<td>SID[1:0]</td>
<td>NSNID[1:0]</td>
<td>NSID[1:0]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for non-secure invasive debug
0x0: Not implemented
ETF device configuration register (ETF_DEVID)

Address offset: 0xFC8
Reset value: 0x0000 01C0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 MEMWIDTH[2:0]: Memory interface data bus width
0x3: 64 bits (corresponds to 32-bit ATB data)

Bits 7:6 CONFIGTYPE[1:0]: Configuration type of component (ETB, ETR or ETF)
0x2: ETF

Bit 5 CLKSCHEME: RAM clocking scheme (synchronous or asynchronous)
0: Synchronous

Bits 4:0 ATBINPORTCNT[4:0]: Number/type of ATB input port multiplexing
0x0: None

ETF device type identifier register (ETF_DEVTYPE)

Address offset: 0xFCC
Reset value: 0x0000 0032

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4 SUBTYPE[3:0]: Sub-classification
0x3: Captures the ATB slave interface trace data in RAM which can then be drained through the ATB master interface

Bits 3:0 MAJORType[3:0]: Major classification
0x2: The component is a trace link because it has an ATB master interface through which trace data can be drained out in Hardware FIFO mode.
ETF CoreSight peripheral identity register 4 (ETF_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

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

Bits 7:4  **SIZE[3:0]**: Register file size
- 0x0: Register file occupies a single 4 Kbyte region

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

ETF CoreSight peripheral identity register 0 (ETF_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0061

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

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]
- 0x61: ETF part number

ETF CoreSight peripheral identity register 1 (ETF_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9

Bits 31:8  Reserved, must be kept at reset value.
ETF CoreSight peripheral identity register 2 (ETF_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 001B

| Bit 31:8 | Reserved, must be kept at reset value. |
| Bit 7:4 | JEP106ID[3:0]: JEP106 identity code field, bits [3:0] |
| 0xB: Arm® JEDEC code |
| Bit 3:0 | PARTNUM[11:8]: Part number field, bits [11:8] |
| 0x9: ETF part number |

ETF CoreSight peripheral identity register 3 (ETF_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

| Bit 31:8 | Reserved, must be kept at reset value. |
| Bit 7:4 | REVAND[3:0]: Metal fix version |
| 0x0: No metal fix |
| Bit 3:0 | CMOD[3:0]: Customer modified |
| 0x0: No customer modifications |
ETF CoreSight component identity register 0 (ETF_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

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

Bits 7:0  PREAMBLE[7:0]: Component ID field, bits [7:0]  
0x0D: Common ID value

ETF CoreSight component identity register 1 (ETF_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

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

Bits 7:4  CLASS[3:0]: Component ID field, bits [15:12] - component class  
0x9: CoreSight component

0x0: Common ID value

ETF CoreSight component identity register 2 (ETF_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

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

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

Bits 7:4  CLASS[3:0]: Component ID field, bits [15:12] - component class  
0x9: CoreSight component

Bits 3:0  PREAMBLE[19:12]: Component ID field, bits [11:8]  
0x0: Common ID value
ETF CoreSight component identity register 3 (ETF_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

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

Bits 7:0 \textbf{PREAMBLE}[19:12]: Component ID field, bits [23:16]
\[0x05: \text{Common ID value}\]

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

Bits 7:0 \textbf{PREAMBLE}[27:20]: Component ID field, bits [31:24]
\[0xB1: \text{Common ID value}\]
### 66.6.12 ETF register map and reset values

#### Table 781. ETF register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>ETF_RSZ</td>
<td>0x0000000100000000000000000000000000000000</td>
<td>RSZ[30:0]</td>
</tr>
<tr>
<td>0x008</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>ETF_STS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>ETF_RRD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>ETF_RRP</td>
<td>0x00000000000000000000000000000000</td>
<td>RRP[12:0]</td>
</tr>
<tr>
<td>0x018</td>
<td>ETF_RWP</td>
<td>0x00000000000000000000000000000000</td>
<td>RWP[12:0]</td>
</tr>
<tr>
<td>0x01C</td>
<td>ETF_TRG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>ETF_CTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x024</td>
<td>ETF_RWD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x028</td>
<td>ETF_MODE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x02C</td>
<td>ETF_LBUFLVL</td>
<td></td>
<td>LBUFLEVEL[11:0]</td>
</tr>
<tr>
<td>0x030</td>
<td>ETF_CBUFLVL</td>
<td></td>
<td>CBUFLEVEL[11:0]</td>
</tr>
<tr>
<td>0x034</td>
<td>ETF_BUFWM</td>
<td></td>
<td>BUFWM[10:0]</td>
</tr>
<tr>
<td>0x038 to 0x2FC</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x300</td>
<td>ETF_FFSR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 781. ETF register map and reset values (continued)

| Offset | Register name  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x304  | ETF_FFCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x308  | ETF_PSCR       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30C  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30D  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x30E  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA0  | ETF_CLAIMSET   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA4  | ETF_CLAIMCLR   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFA8  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB0  | ETF_LAR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | ACCESS_W[31:0] |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB4  | ETF_LSR        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFB8  | ETF_AUTHSTAT   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFBC  | Reserved       |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC8  | ETF_DEVID      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

#### Register Description

- **ETF_FFCR**: Debug Infrastructure RM0477
- **ETF_PSCR**: Reset value
- **ETF_CLAIMSET**: Reset value
- **ETF_CLAIMCLR**: Reset value
- **ETF_LAR**: ACCESS_W[31:0]
- **ETF_LSR**: LockGrant, LockType
- **ETF_AUTHSTAT**: LockExist, SNID[1:0], SID[1:0], NSNID[1:0], NSID[1:0]
- **ETF_DEVID**: MEMWDTH[2:0], ATBINPORTCNT[4:0]
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>ETF_DEVTYPE</td>
<td>SUBTYPE[3:0]</td>
<td>MAJOR_TYPE[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 1 1 0 0 1 0</td>
</tr>
<tr>
<td>0xFD0</td>
<td>ETF_PIDR4</td>
<td>RES[9]</td>
<td>JEP106CONF[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0xFD4</td>
<td>ETF_PIDR3</td>
<td>PARTNUM[7:0]</td>
<td>JEP106CONF[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0xFF0</td>
<td>ETF_CIDR0</td>
<td>PREAMBLE[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 1 1 0 1</td>
</tr>
<tr>
<td>0xFF4</td>
<td>ETF_CIDR1</td>
<td>CLASS[3:0]</td>
<td>PREAMBLE[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0 0 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0xFF8</td>
<td>ETF_CIDR2</td>
<td></td>
<td>PREAMBLE[19:12]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0 0 0 0 0 1 0 0 1</td>
</tr>
<tr>
<td>0xFFC</td>
<td>ETF_CIDR3</td>
<td></td>
<td>PREAMBLE[27:20]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1 0 1 1 0 0 0 0 1</td>
</tr>
</tbody>
</table>
66.6.13 **Trace port interface unit (TPIU)**

The TPIU is a CoreSight™ component that formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECK, and four data outputs, TRACED(3:0). The trace port width is programmable in the range 1 to 4. Using a smaller port width reduces the number of test points/connector pins needed, and frees up IOs for other purposes. However it restricts the bandwidth of the trace port and hence the quantity of trace information that can be output in real time. The ATB trace bus must be enabled by setting the TRACECLKEN bit in the DBGMCU_CR register before a trace is sent to the TPIU.

Trace data is output on TRACED[3:0] synchronously with the rising and falling edges of TRACECK. TRACECK is synchronous with the TRACECLKIN input, at half the frequency. This means that TRACECK is asynchronous with respect to the ATB clock. The asynchronism is handled by a FIFO in the TPIU, which also regulates the flow of data from the ETF. If the trace port bandwidth is not sufficient to handle the amount of trace data, the ETF can overflow and trace data can be missed. In this case the TRACECLKIN frequency can be increased, or the port size increased, up to a maximum of four pins. Alternatively, the ETM can be programmed to filter unnecessary information.

For more information on the Trace port interface CoreSight™ component, refer to the Arm® CoreSight™ SoC-400 technical reference manual [2].
### 66.6.14 TPIU registers

**TPIU supported port size register (TPIU_SUPPSIZE)**

Address offset: 0x000  
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TPIU current port size register (TPIU_CURPSIZE)**

Address offset: 0x004  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**TPIU supported trigger modes register (TPIU_SUPTRGM)**

Address offset: 0x100  
Reset value: 0x0000 011F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>TRGR</th>
<th>TRIGD</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>TCOU</td>
<td>N TB</td>
<td>MULT6</td>
<td>4K</td>
<td>MULT2</td>
<td>56</td>
<td>MULT1</td>
<td>6</td>
<td>MULT4</td>
<td>MULT2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **PORTSIZE[31:0]**: Indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.  
0x0000 000F: Port sizes 1 to 4 supported

Bits 31:0 **PORTSIZE[31:0]**: Current trace port size  
Setting bit n-1 indicates that the current port size is n-pin wide. The value of n must be within the range of supported port sizes (1-4). Only one bit can be set, or unpredictable behavior may result. This register should only be modified when the formatter is stopped.
Bits 31:18  Reserved, must be kept at reset value.

Bit 17  **TRGRUN**: Trigger running
0: Trigger has not occurred or counter is at 0
1: Trigger has occurred and counter is not at 0

Bit 16  **TRIGD**: Triggered
0: Trigger has not occurred
1: Trigger has occurred and counter has reached 0

Bits 15:9  Reserved, must be kept at reset value.

Bit 8  **TCOUNT8**: 8-bit counter register
1: Implemented

Bits 7:5  Reserved, must be kept at reset value.

Bit 4  **MULT64K**: Multiplying the trigger counter by 65536 support
1: Supported

Bit 3  **MULT256**: Multiplying the trigger counter by 256 support
1: Supported

Bit 2  **MULT16**: Multiplying the trigger counter by 16 support
1: Supported

Bit 1  **MULT4**: Multiplying the trigger counter by 4 support
1: Supported

Bit 0  **MULT2**: Multiplying the trigger counter by 2 support
1: Supported

**TPIU trigger counter value register (TPIU_TRGCNT)**

Address offset: 0x104

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **TRIGCOUNT[7:0]**: Enable trigger delay indication

Enables delaying the indication of triggers to any externally connected trace capture or storage devices. This counter is only eight bits wide and is intended to be used only with the counter multipliers in the Trigger multiplier register, 0x108. When a trigger is started, this value, in conjunction with the multiplier, specifies the number of words before the trigger is indicated. When the trigger counter reaches 0, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but does not reset any value on the multiplier. Reading this register returns the preset value, not the current count.
### TPIU Trigger Multiplier Register (TPIU_TRGMULT)

**Address offset:** 0x108

**Reset value:** 0x0000 0000

| Bit 31-5 | Reserved, must be kept at reset value. |
| Bit 4   | **MULT64K:** Multiply the trigger counter by 65536 |
|         | 0: Disabled |
|         | 1: Enabled |
| Bit 3   | **MULT256:** Multiply the trigger counter by 256 |
|         | 0: Disabled |
|         | 1: Enabled |
| Bit 2   | **MULT16:** Multiply the trigger counter by 16 |
|         | 0: Disabled |
|         | 1: Enabled |
| Bit 1   | **MULT4:** Multiply the trigger counter by 4 |
|         | 0: Disabled |
|         | 1: Enabled |
| Bit 0   | **MULT2:** Multiply the trigger counter by 2 |
|         | 0: Disabled |
|         | 1: Enabled |

### TPIU Supported Test Patterns/Modes Register (TPIU_SUPTPM)

**Address offset:** 0x200

**Reset value:** 0x0000 000F

| Bit 31-5 | Reserved, must be kept at reset value. |
| Bit 24-20 | PCONT, EN, PTIME |
| Bit 19-15 | PATW0, PATW1 |
| Bit 14-10 | PATA5, PATF0 |
| Bit 9-5  | MULT2, MULT4, MULT1, MULT6 |
| Bit 4-0  | MULT2, MULT4, MULT1, MULT64K |

| Bit 31-5 | Reserved, must be kept at reset value. |
| Bit 24-20 | PCONT, EN, PTIME |
| Bit 19-15 | PATW0, PATW1 |
| Bit 14-10 | PATA5, PATF0 |
| Bit 9-5  | MULT2, MULT4, MULT1, MULT6 |
| Bit 4-0  | MULT2, MULT4, MULT1, MULT64K |
### TPIU current test pattern/mode register (TPIU_CURTPM)

**Address offset:** 0x204

**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PATF0</td>
<td>PATA5</td>
<td>PATW0</td>
<td>PATW1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bit 17** **PCONTEN**: Continuous mode enable
  - 0: Disabled
  - 1: Enabled

- **Bit 16** **PTIMEEN**: Timed mode enable
  - 0: Disabled
  - 1: Enabled

#### Bits 15:4 Reserved, must be kept at reset value.

- **Bit 3** **PATF0**: FF/00 pattern enable
  - Indicates whether the FF/00 pattern is supported as output over the trace port.
  - 0: Disabled
  - 1: Enabled
Bit 2 **PATA5**: AA/55 pattern is enable
- Indicates whether the AA/55 pattern is enabled as output over the trace port
  - 0: Disabled
  - 1: Enabled

Bit 1 **PATW0**: Walking 0’s pattern enable
- Indicates whether the walking 0’s pattern is enabled as output over the trace port
  - 0: Disabled
  - 1: Enabled

Bit 0 **PATW1**: Walking 1’s pattern enable
- Indicates whether the walking 1’s pattern is enabled as output over the trace port
  - 0: Disabled
  - 1: Enabled

**TPIU test pattern repeat counter register (TPIU_TPRCR)**

Address offset: 0x208
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>PATTCOUNT[7:0]</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PATTCOUNT[7:0]**: Number of TRACECLKIN cycles
- The field provides a 8-bit counter value to indicate the number of TRACECLKIN cycles for which a pattern runs before it switches to the next pattern.

**TPIU formatter and flush status register (TPIU_FFSR)**

Address offset: 0x300
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>Bit 31-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>TCPRESET/FTSTOP/FLINP</td>
</tr>
</tbody>
</table>

- r: read
- w: write
Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TCPRESENT**: TRACECTL output pin availability
Indicates whether the optional TRACECTL output pin is available for use.
0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: Formatter stopped
The formatter has received a stop request signal and all trace data and post-amble is sent.
Any additional trace data on the ATB interface is ignored.
0: Formatter has not stopped
1: Formatter has stopped

Bit 0 **FLINPROG**: Flush in progress
Indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in the TPIU_FFCR register.
0: No flush in progress
1: Flush in progress

**TPIU formatter and flush control register (TPIU_FFCR)**
Address offset: 0x304
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STOPTRIG**: Stop on trigger event
0: No effect
1: Stop formatter when a trigger event occurs

Bit 12 **STOPFL**: Stop on flush
0: No effect
1: Stop formatter when flush is completed

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGFL**: Trigger on flush
0: No effect
1: Indicate a trigger in the trace stream when flush is completed

Bit 9 **TRIGEVT**: Trigger on trigger event
0: No effect
1: Indicate a trigger in the trace stream when trigger event occurs

Bit 8 **TRIGIN**: Trigger on trigger in
0: No effect
1: Indicate a trigger in the trace stream when the TRIGIN input from the system CTI is asserted.
Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: Generate a manual flush
   0: No effect
   1: Flush the trace
   This bit is cleared automatically when the flush completes.

Bit 5 **FONTRIG**: Flush on trigger event
   A trigger event occurs when the trigger counter reaches 0, or, if the trigger counter is 0, when the TRIGIN input from the system CTI is high.
   0: No effect
   1: Flush the trace if a trigger event occurs

Bit 4 **FONFLIN**: Flush on flush in
   0: No effect
   1: Flush the trace if the FLUSHIN input from the system CTI is asserted

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT**: Enable continuous formatting
   0: Continuous formatting is disabled
   1: Continuous formatting is enabled

Bit 0 **ENFTC**: Enable the embedding of triggers in formatted trace
   0: Formatting is disabled
   1: Formatting is enabled

**TPIU formatter synchronization counter register (TPIU_FSCR)**

Address offset: 0x400
Reset value: 0x0000 0040

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

31:12 Reserved, must be kept at reset value.

**Bits 11:0 CYCCOUNT[11:0]**: Enables effective use of TPAs
   Enables effective use of different-sized TPAs without wasting large amounts of storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to a synchronization every 65536 bytes, that is, 4096 packets x 16 bytes per packet. The default is set up for a synchronization packet every 1024 bytes, that is, every 64 formatter frames. If the formatter is configured for continuous mode, full and half-word sync frames are inserted during normal operation. Under these circumstances, the count value is the maximum number of complete frames between full synchronization packets.
### TPIU claim tag set register (TPIU_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0  **CLAIMSET[3:0]:** Set claim tag bits

Write:
- 0000: No effect
- xxx1: Set bit 0
- xx1x: Set bit 1
- x1xx: Set bit 2
- 1xxx: Set bit 3

Read:
- 0xF: Indicates there are four bits in claim tag

### TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0  **CLAIMCLR[3:0]:** Reset claim tag bits

Write:
- 0000: No effect
- xxx1: Clear bit 0
- xx1x: Clear bit 1
- x1xx: Clear bit 2
- 1xxx: Clear bit 3

Read: Returns current value of claim tag
TPIU lock access register (TPIU_LAR)

Address offset: 0xFB0
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:0</td>
<td>ACCESS_W[31:0]: TPIU register access enable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Enables write access to some TPIU registers by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the processor cores (debuggers do not need</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to unlock the component)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0xC5ACCE55: Enable write access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Other values: Disable write access</td>
<td></td>
</tr>
</tbody>
</table>

TPIU lock status register (TPIU_LSR)

Address offset: 0xFB4
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:3</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 2 LOCKTYPE: Size of the TPIU_LAR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>register</td>
<td>0: 32-bit</td>
</tr>
<tr>
<td></td>
<td>Bit 1 LOCKGRANT: Current status of lock</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This bit always returns zero when read by</td>
<td></td>
</tr>
<tr>
<td></td>
<td>an external debugger.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: Write access is permitted</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Write access is blocked. Only read</td>
<td></td>
</tr>
<tr>
<td></td>
<td>access is permitted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 0 LOCKEXIST: Implementation of a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>lock control mechanism</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The bit indicates whether a lock control</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mechanism is implemented. It always</td>
<td></td>
</tr>
<tr>
<td></td>
<td>returns zero when read by an external</td>
<td></td>
</tr>
<tr>
<td></td>
<td>debugger.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0: No lock control mechanism is available</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1: Lock control mechanism is implemented</td>
<td></td>
</tr>
</tbody>
</table>
### TPIU authentication status register (TPIU_AUTHSTAT)

Address offset: 0xFB8  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:6  **SNID[1:0]**: Security level for secure non-invasive debug  
0x0: Not implemented

Bits 5:4  **SID[1:0]**: Security level for secure invasive debug  
0x0: Not implemented

Bits 3:2  **NSNID[1:0]**: Security level for non-secure non-invasive debug  
0x0: Not implemented

Bits 1:0  **NSID[1:0]**: Security level for non-secure invasive debug  
0x0: Not implemented

### TPIU device configuration register (TPIU_DEVID)

Address offset: 0xFC8  
Reset value: 0x0000 00A0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:12  Reserved, must be kept at reset value.

Bit 11  **SWOUARTNRZ**: Support of SWO UART or NRZ  
Indicates whether serial wire output, UART or NRZ, is supported.

0: Not supported

Bit 10  **SWOMAN**: Support of SWO Manchester format  
Indicates whether serial wire output, Manchester encoded format, is supported.

0: Not supported

Bit 9  **TCLKDATA**: Support of trace clock plus data  
0: Not supported
Bits 8:6  **FIFOSIZE[2:0]**: FIFO size in powers of 2
0x2: FIFO size = 4 (16 bytes)

Bit 5  **CLKRELAT**: ATB clock and TRACECLKIN relation
Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)
1: Asynchronous

Bits 4:0  **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

**TPIU device type identifier register (TPIU_DEVTYPE)**
Address offset: 0xFCC
Reset value: 0x0000 0011

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBTYPE[3:0]</td>
<td>MAJORTYPE[3:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0  **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

**TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)**
Address offset: 0xFD0
Reset value: 0x0000 0004

<p>| | | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>r</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code
### TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0012

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PARTNUM[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]
0x12: TPIU part number

### TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r r</td>
</tr>
</tbody>
</table>

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

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

Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: TPIU part number

### TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 005B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r r r</td>
</tr>
</tbody>
</table>

3662/3791  RM0477 Rev 6
RM0477 Debug infrastructure

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

Bits 7:4  **REVISION[3:0]**: Component revision number
          0x5: r1p0

Bit 3  **JEDEC**: JEDEC assigned value
       1: Designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
          0x3: Arm® JEDEC code

**TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)**

Address offset: 0xFEC
Reset value: 0x0000 0000

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REVAND[3:0]</th>
<th>CMOD[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
```

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

Bits 7:4  **REVAND[3:0]**: Metal fix version
          0x0: No metal fix

Bits 3:0  **CMOD[3:0]**: Customer modified
          0x0: No customer modifications

**TPIU CoreSight component identity register 0 (TPIU_CIDR0)**

Address offset: 0xFF0
Reset value: 0x0000 000D

```
<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PREAMBLE[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
</tr>
<tr>
<td>r</td>
</tr>
<tr>
<td>r</td>
</tr>
<tr>
<td>r</td>
</tr>
</tbody>
</table>
```

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

Bits 7:0  **PREAMBLE[7:0]**: Component ID field, bits [7:0]
          0x0D: Common ID value
TPIU CoreSight component identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

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

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [23:16]
0x05: Common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]

0xB1: Common ID value
## 66.6.15 TPIU register map and reset values

### Table 782. TPIU register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset value</th>
<th>Name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>TPIU_SUPPSIZE</td>
<td>0x000</td>
<td>PORTSIZE[31:0]</td>
<td>0x00000000000000000000000011111111</td>
</tr>
<tr>
<td>0x004</td>
<td>TPIU_CURPSIZE</td>
<td>0x004</td>
<td>PORTSIZE[31:0]</td>
<td>0x00000000000000000000000000000001</td>
</tr>
<tr>
<td>0x008</td>
<td>Reserved</td>
<td>0x008</td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x010</td>
<td>TPIU_SUPTRGM</td>
<td>0x010</td>
<td>TRGMEN</td>
<td>0x00000000000000000000000011111111</td>
</tr>
<tr>
<td>0x014</td>
<td>TPIU_TRGCNT</td>
<td>0x014</td>
<td>TRGCNT[7:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x018</td>
<td>TPIU_TRGMULT</td>
<td>0x018</td>
<td>MULT64K, MULT256, MULT16, MULT4, MULT2</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x01C</td>
<td>Reserved</td>
<td>0x01C</td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x020</td>
<td>TPIU_SUPTPM</td>
<td>0x020</td>
<td>PCONTEN, PTIMEEN</td>
<td>0x00000000000000000000000011111111</td>
</tr>
<tr>
<td>0x024</td>
<td>TPIU_CURTPM</td>
<td>0x024</td>
<td>PCONTEN, PTIMEEN</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x028</td>
<td>TPIU_TPRCR</td>
<td>0x028</td>
<td>PATTCOUNT[7:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x02C</td>
<td>Reserved</td>
<td>0x02C</td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x030</td>
<td>TPIU_FFSR</td>
<td>0x030</td>
<td>STOPTRIG, STOPFL, TRGFL, TRGEGN, TRGIN</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x034</td>
<td>TPIU_FFCR</td>
<td>0x034</td>
<td>FONMAN, FONTRIG, FONFLIN, PCNT, ENFCONT, ENFTC</td>
<td>0x000000000000000000000000</td>
</tr>
<tr>
<td>0x038</td>
<td>TPIU_FSCR</td>
<td>0x038</td>
<td>CYCCOUNT[11:0]</td>
<td>0x00000000000000000000000000000000</td>
</tr>
<tr>
<td>0x03C</td>
<td>Reserved</td>
<td>0x03C</td>
<td>Reserved</td>
<td>0x00000000000000000000000000000000</td>
</tr>
</tbody>
</table>
### Table 782. TPIU register map and reset values (continued)

| Offset  | Register name            | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|--------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFA0   | TPIU_CLAIMSET            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFA4   | TPIU_CLAIMCLR            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFB0   | TPIU_LAR                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFB4   | TPIU_LSR                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFB8   | TPIU_AUTHSTAT            |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFBC   | Reserved                 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFC8   | TPIU_DEVID               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFD0   | TPIU_PIDR4               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFE0   | TPIU_PIDR0               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFE4   | TPIU_PIDR1               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
| 0xFE8   | TPIU_PIDR2               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|         | Reset value              | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |    |    |    |    |    |
66.7 Serial wire output (SWO)

The SWO is a CoreSight component that formats the trace stream from the processor ITM and outputs it on the single wire TRACESWO.

Compared to the TPIU, the SWO contains:
- no formatter
- no pattern generator
- an 8-bit ATB input
- no synchronous trace output, that is, no TRACEDATA or TRACECLK pins
- no support for flush, because this is not required
- no support for triggering

The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the serial wire output CoreSight™ component, refer to the Arm® CoreSight™ Components Technical Reference Manual [4].
### SWO registers

#### SWO current output divisor register (SWO_CODR)

Address offset: 0x010  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:13  Reserved, must be kept at reset value.

Bits 12:0  **PRESCALER[12:0]**: SWO baud rate scaling  
The baud rate is the trace clock frequency divided by (PRESCALER - 1). The baud rate changes instantly, so it is recommended to stop the trace source and wait until the port is idle before writing to this register.

#### SWO selected pin protocol register (SWO_SPPR)

Address offset: 0x0F0  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 1:0  **PPROT[1:0]**: Pin protocol  
0x0: Reserved  
0x1: Manchester  
0x2: NRZ  
0x3: Reserved
### SWO formatter and flush status register (SWO_FFSR)

Address offset: 0x300  
Reset value: 0x0000 0008

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FTNONSTOP</th>
<th>TCPRESENT</th>
<th>FTSTOPPED</th>
<th>FLINPROG</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

- **Bit 3** FTNONSTOP: Change of settings without stopping formatter
  1: Change of settings is allowed with formatter running

- **Bit 2** TCPRESENT: TRACECTL pin present on SWO
  0: TRACECTL pin not present

- **Bit 1** FTSTOPPED: Formatter stopped
  0: Formatter running
  The bit always returns 0 as the SWO formatter cannot be stopped in this device.

- **Bit 0** FLINPROG: Flush in progress
  0: Flush is not in progress
  The bit always returns 0 as SWO flushing is not supported in this device.

### SWO claim tag set register (SWO_CLAIMSET)

Address offset: 0xFA0  
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLAIMSET[3:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
</tr>
</tbody>
</table>

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

- **Bits 3:0** CLAIMSET[3:0]: Set claim tag bits
  - **Write:**
    - 0000: No effect
    - xxx1: Set bit 0
    - xx1x: Set bit 1
    - x1xx: Set bit 2
    - 1xxx: Set bit 3
  - **Read:**
    - 0xF: Indicates there are four bits in claim tag
**SWO claim tag clear register (SWO_CLAIMCLR)**

Address offset: 0xFA4  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.  
Bits 3:0  **CLAIMCLR[3:0]**: Reset claim tag bits  
Write:  
0000: No effect  
xxx1: Clear bit 0  
xx1x: Clear bit 1  
x1xx: Clear bit 2  
1xxx: Clear bit 3  
Read: Returns current value of claim tag

**SWO lock access register (SWO_LAR)**

Address offset: 0xFB0  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **ACCESS_W[31:0]**: SWO register write access enable  
Enables write access to some SWO registers by the processor cores (debuggers do not need to unlock the component)  
0xC5ACCE55: Enable write access  
Other values: Disable write access
### SWO lock status register (SWO_LSR)

**Address offset:** 0xFB4  
**Reset value:** 0x0000 0003

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:3</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><strong>LOCKTYPE:</strong> Size of the SWO_LAR register</td>
<td>0: 32-bit</td>
</tr>
</tbody>
</table>
| 1    | **LOCKGRANT:** Current status of lock             | 0: Write access is permitted  
|      |                                                   | 1: Write access is blocked - only read access is permitted |
| 0    | **LOCKEXIST:** Implementation of a lock control mechanism | 0: No lock control mechanism available  
|      |                                                   | 1: Lock control mechanism is implemented |

### SWO authentication status register (SWO_AUTHSTAT)

**Address offset:** 0xFB8  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>31:8</td>
<td>Reserved, must be kept at reset value</td>
<td></td>
</tr>
</tbody>
</table>
| 8    | **SNID[1:0]:** Security level for secure non-invasive debug  
|      | 0x0: Not implemented                             |         |
| 7:6  | **SID[1:0]:** Security level for secure invasive debug  
|      | 0x0: Not implemented                             |         |
| 5:4  | **NSNID[1:0]:** Security level for non-secure non-invasive debug  
|      | 0x0: Not implemented                             |         |
| 3:2  | **NSID[1:0]:** Security level for non-secure invasive debug  
|      | 0x0: Not implemented                             |         |
**SWO device configuration register (SWO_DEVID)**

Address offset: 0xFC8

Reset value: 0x0000 0EA0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWO_UART_NRZ</td>
<td>SWO_MAN</td>
<td>TCLK_DATA</td>
<td>FIFO_SIZE[2:0]</td>
<td>CLK_RELAT</td>
<td>MUXNUM[4:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:12 Reserved, must be kept at reset value.

- **Bit 11 SWO_UART_NRZ**: SWO UART or NRZ support
  Indicates whether serial wire output, UART or NRZ, is supported.
  1: Supported

- **Bit 10 SWO_MAN**: SWO Manchester format support
  Indicates whether serial wire output, Manchester encoded format, is supported.
  1: Supported

- **Bit 9 TCLK_DATA**: Trace clock plus data support
  Indicates whether trace clock plus data is supported.
  1: Supported

- **Bits 8:6 FIFO_SIZE[2:0]**: FIFO size in powers of 2
  0x2: FIFO size = 4 (16 bytes)

- **Bit 5 CLK_RELAT**: ATB clock to TRACECLKIN relation
  Indicates the relationship between the ATB clock and TRACECLKIN (synchronous or asynchronous)
  1: Asynchronous

- **Bits 4:0 MUXNUM[4:0]**: Number/type of ATB input port multiplexing
  0x0: None

**SWO device type identifier register (SWO_DEVTYPE)**

Address offset: 0xFCC

Reset value: 0x0000 0011

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBTYPE[3:0]</td>
<td>MAJOR_TYPE[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0  **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

**SWO CoreSight peripheral identity register 4 (SWO_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SWO CoreSight peripheral identity register 0 (SWO_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

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

**SWO CoreSight peripheral identity register 4 (SWO_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0  **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

**SWO CoreSight peripheral identity register 4 (SWO_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SWO CoreSight peripheral identity register 0 (SWO_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

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

**SWO CoreSight peripheral identity register 4 (SWO_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SWO CoreSight peripheral identity register 0 (SWO_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

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

**SWO CoreSight peripheral identity register 4 (SWO_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SWO CoreSight peripheral identity register 0 (SWO_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 0014

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

Bits 3:0  **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm® JEDEC code
**SWO CoreSight peripheral identity register 1 (SWO_PIDR1)**

Address offset: 0xFE4
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Bits 31:8  Reserved, must be kept at reset value.
- Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
  - 0xB: Arm® JEDEC code
- Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]
  - 0x9: SWO part number

**SWO CoreSight peripheral identity register 2 (SWO_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 002B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Bits 31:8  Reserved, must be kept at reset value.
- Bits 7:4  **REVISION[3:0]**: Component revision number
  - 0x2: r0p2
- Bit 3  **JEDEC**: JEDEC assigned value
  - 1: Designer ID specified by JEDEC
- Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
  - 0x3: Arm® JEDEC code
**SWO CoreSight peripheral identity register 3 (SWO_PIDR3)**

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

---

**SWO CoreSight component identity register 0 (SWO_CIDR0)**

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0xD: Common ID value

---

**SWO CoreSight component identity register 1 (SWO_CIDR1)**

Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

**SWO CoreSight component identity register 2 (SWO_CIDR2)**

Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

**SWO CoreSight component identity register 3 (SWO_CIDR3)**

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value
### 66.7.2 SWO register map and reset values

Table 783. SWO register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010</td>
<td>SWO_CODR</td>
<td>31</td>
<td>PRESCALER[12:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x014</td>
<td>SWO_SPPR</td>
<td>26</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0F0</td>
<td>SWO_FFSR</td>
<td>24</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0F4</td>
<td>SWO_CLAIMSET</td>
<td>22</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0A0</td>
<td>SWO_CLAIMCLR</td>
<td>20</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0B0</td>
<td>SWO_LAR</td>
<td>18</td>
<td>ACCESS_W[31:0]</td>
</tr>
<tr>
<td>0x0B4</td>
<td>SWO_LSR</td>
<td>16</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0B8</td>
<td>SWO_AUTHSTAT</td>
<td>14</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x0C0</td>
<td>SWO_ID</td>
<td>12</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x010 SWO_CODR

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010</td>
<td>SWO_CODR</td>
<td>31</td>
<td>PRESCALER[12:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0F0 SWO_SPPR

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0F0</td>
<td>SWO_SPPR</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0F4 SWO_FFSR

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0F4</td>
<td>SWO_FFSR</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0FA SWO_CLAIMSET

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FA</td>
<td>SWO_CLAIMSET</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0FA SWO_CLAIMCLR

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FA</td>
<td>SWO_CLAIMCLR</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0FB SWO_LAR

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FB</td>
<td>SWO_LAR</td>
<td>31</td>
<td>ACCESS_W[31:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0FB SWO_LSR

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FB</td>
<td>SWO_LSR</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0FB SWO_AUTHSTAT

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FB</td>
<td>SWO_AUTHSTAT</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

#### Offset 0x0FC SWO_ID

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0FC</td>
<td>SWO_ID</td>
<td>31</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>28</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
Table 783. SWO register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFC8</td>
<td>SWO_DEVID</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFCC</td>
<td>SWO_DEVTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SUBTYPE[3:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0xFD0</td>
<td>SWO_PIDR4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0xFD4 to</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>SWO_PIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PARTNUM[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0xFE4</td>
<td>SWO_PIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>JEP106CON[3:0]</td>
<td>PARTNUM[1:8]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>SWO_PIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>REVISION[3:0]</td>
<td>JEP106ID[6:4]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0xFE8</td>
<td>SWO_PIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>REVAND[3:0]</td>
<td>CMOD[3:0]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFF0</td>
<td>SWO_CIDR0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[7:0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFF4</td>
<td>SWO_CIDR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CLASS[3:0]</td>
<td>PREAMBLE[11:8]</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0xFF8</td>
<td>SWO_CIDR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[19:12]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xFFC</td>
<td>SWO_CIDR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[27:20]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
66.8  Microcontroller debug unit (DBGMCU)

The DBGMCU component contains a number of registers that control the power and clock behavior in Debug mode. Specifically it allows the debugger, or debug software, to:

- maintain the clock and power to the processor cores when in low-power modes (sleep, stop or standby)
- maintain the clock and power to the system debug and trace components when in low power modes
- stop the clock to certain peripherals (CAN, SMBUS timeout, Watchdogs, Timers, RTC) when the processor core is stopped in Debug mode. For timers having complementary outputs, these outputs are disabled (as if the MOE bit was reset) for safety purposes when the counter is stopped (TIM1/15/16/17 = 1 in DBGMCU_APB2FZR).

The DBGMCU registers are not reset by a system reset, only by a power on reset. They are accessible to the debugger via the APB-D bus at base address 0xE00E1000. They are also accessible by the processor core at base address 0x5C001000.
66.8.1 DBGMCU registers

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000
Reset value: 0x1003 6485

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16 \( \text{REV}_\text{ID}[15:0] \): 0x1003 = revision Y

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

Bits 11:0 \( \text{DEV}_\text{ID}[11:0] \): Device ID
0x485: STM32H7Rx/7Sx

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 \( \text{TRGOEN} \): External trigger output enable
This bit controls the direction of the bi-directional trigger pin, TRGIO.
0: Input - TRGIO is connected to TRGIN
1: Output - TRGIO is connected to TRGOUT

Bits 27:22 Reserved, must be kept at reset value.

Bit 21 \( \text{DBGCKEN} \): debug clock enable
This bit allows clocks of the debug components (excluding those in the processor core) to be switched off if they are not needed.
0: Disabled - clock debug components are disabled and their clocks gated
1: Enabled - clock debug components are clocked whenever the corresponding domain clock (CK_HCLK) is active
Bit 20  **TRACECLKEN**: Trace port clock enable.
This bit enables the trace port, TRACECLK.
0: Disabled - TRACECLK is disabled
1: Enabled - TRACECLK is active

Bits 19:17  Reserved, must be kept at reset value.

Bit 16  **DCRT**: Debug credentials reset type
This bit selects which type of reset is used to revoke the debug authentication credentials
0: System reset
1: Power reset

Bits 15:3  Reserved, must be kept at reset value.

Bit 2  **DBG_STANDBY**: Debug in Standby mode enable
0: Normal operation - all clocks are disabled and the core powered down automatically in Standby mode.
1: Automatic clock stop/power-down disabled - all active clocks and oscillators continue to run during Standby mode, and the core supply is maintained, allowing full debug capability. On exit from Standby mode, a core reset is performed.

Bit 1  **DBG_STOP**: Debug in Stop mode enable
0: Normal operation - all clocks are disabled automatically in Stop mode
1: Automatic clock stop disabled - all active clocks and oscillators continue to run during Stop mode, allowing full debug capability. On exit from Stop mode, the clock settings are set to the Stop mode exit state.

Bit 0  **DBG_SLEEP**: Debug in Sleep mode enable
0: Normal operation - the processor clock is stopped automatically in Sleep mode
1: Automatic clock stop disabled - processor clock continues to run, allowing full debug capability

**DBGMCU AHB5 peripheral freeze register (DBGMCU_AHB5FZR)**

Address offset: 0x01C
Reset value: 0x0000 0000

Description: This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.
Bits 31:16  Reserved, must be kept at reset value.

Bit 15  **HPDMA_15**: HPDMA channel 15 stop in debug
        0: normal operation. HPDMA channel 15 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 15 is frozen while the CPU is in debug mode.

Bit 14  **HPDMA_14**: HPDMA channel 14 stop in debug
        0: normal operation. HPDMA channel 14 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 14 is frozen while the CPU is in debug mode.

Bit 13  **HPDMA_13**: HPDMA channel 13 stop in debug
        0: normal operation. HPDMA channel 13 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 13 is frozen while the CPU is in debug mode.

Bit 12  **HPDMA_12**: HPDMA channel 12 stop in debug
        0: normal operation. HPDMA channel 12 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 12 is frozen while the CPU is in debug mode.

Bit 11  **HPDMA_11**: HPDMA channel 11 stop in debug
        0: normal operation. HPDMA channel 11 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 11 is frozen while the CPU is in debug mode.

Bit 10  **HPDMA_10**: HPDMA channel 10 stop in debug
        0: normal operation. HPDMA channel 10 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 10 is frozen while the CPU is in debug mode.

Bit 9   **HPDMA_9**: HPDMA channel 9 stop in debug
        0: normal operation. HPDMA channel 9 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 9 is frozen while the CPU is in debug mode.

Bit 8   **HPDMA_8**: HPDMA channel 8 stop in debug
        0: normal operation. HPDMA channel 8 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 8 is frozen while the CPU is in debug mode.

Bit 7   **HPDMA_7**: HPDMA channel 7 stop in debug
        0: normal operation. HPDMA channel 7 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 7 is frozen while the CPU is in debug mode.

Bit 6   **HPDMA_6**: HPDMA channel 6 stop in debug
        0: normal operation. HPDMA channel 6 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 6 is frozen while the CPU is in debug mode.

Bit 5   **HPDMA_5**: HPDMA channel 5 stop in debug
        0: normal operation. HPDMA channel 5 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 5 is frozen while the CPU is in debug mode.

Bit 4   **HPDMA_4**: HPDMA channel 4 stop in debug
        0: normal operation. HPDMA channel 4 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 4 is frozen while the CPU is in debug mode.

Bit 3   **HPDMA_3**: HPDMA channel 3 stop in debug
        0: normal operation. HPDMA channel 3 continues to operate while the CPU is in debug mode.
        1: stop in debug. HPDMA channel 3 is frozen while the CPU is in debug mode.
Bit 2  **HPDMA_2**: HPDMA channel 2 stop in debug
   0: normal operation. HPDMA channel 2 continues to operate while the CPU is in debug mode.
   1: stop in debug. HPDMA channel 2 is frozen while the CPU is in debug mode.

Bit 1  **HPDMA_1**: HPDMA channel 1 stop in debug
   0: normal operation. HPDMA channel 1 continues to operate while the CPU is in debug mode.
   1: stop in debug. HPDMA channel 1 is frozen while the CPU is in debug mode.

Bit 0  **HPDMA_0**: HPDMA channel 0 stop in debug
   0: normal operation. HPDMA channel 0 continues to operate while the CPU is in debug mode.
   1: stop in debug. HPDMA channel 0 is frozen while the CPU is in debug mode.

**DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)**

Address offset: 0x024

Reset value: 0x0000 0000

Description: This register is accessible to the debugger after successful authentication. Prior to this, debugger accesses are ignored. It is always accessible to software.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:16  Reserved, must be kept at reset value.

Bit 15  **GPDMA_15**: GPDMA channel 15 stop in debug
   0: normal operation. GPDMA channel 15 continues to operate while the CPU is in debug mode.
   1: stop in debug. GPDMA channel 15 is frozen while the CPU is in debug mode.

Bit 14  **GPDMA_14**: GPDMA channel 14 stop in debug
   0: normal operation. GPDMA channel 14 continues to operate while the CPU is in debug mode.
   1: stop in debug. GPDMA channel 14 is frozen while the CPU is in debug mode.

Bit 13  **GPDMA_13**: GPDMA channel 13 stop in debug
   0: normal operation. GPDMA channel 13 continues to operate while the CPU is in debug mode.
   1: stop in debug. GPDMA channel 13 is frozen while the CPU is in debug mode.

Bit 12  **GPDMA_12**: GPDMA channel 12 stop in debug
   0: normal operation. GPDMA channel 12 continues to operate while the CPU is in debug mode.
   1: stop in debug. GPDMA channel 12 is frozen while the CPU is in debug mode.

Bit 11  **GPDMA_11**: GPDMA channel 11 stop in debug
   0: normal operation. GPDMA channel 11 continues to operate while the CPU is in debug mode.
   1: stop in debug. GPDMA channel 11 is frozen while the CPU is in debug mode.

Bit 10  **GPDMA_10**: GPDMA channel 10 stop in debug
   0: normal operation. GPDMA channel 10 continues to operate while the CPU is in debug mode.
   1: stop in debug. GPDMA channel 10 is frozen while the CPU is in debug mode.
Bit 9 **GPDMA_9**: GPDMA channel 9 stop in debug
0: normal operation. GPDMA channel 9 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 9 is frozen while the CPU is in debug mode.

Bit 8 **GPDMA_8**: GPDMA channel 8 stop in debug
0: normal operation. GPDMA channel 8 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 8 is frozen while the CPU is in debug mode.

Bit 7 **GPDMA_7**: GPDMA channel 7 stop in debug
0: normal operation. GPDMA channel 7 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 7 is frozen while the CPU is in debug mode.

Bit 6 **GPDMA_6**: GPDMA channel 6 stop in debug
0: normal operation. GPDMA channel 6 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 6 is frozen while the CPU is in debug mode.

Bit 5 **GPDMA_5**: GPDMA channel 5 stop in debug
0: normal operation. GPDMA channel 5 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 5 is frozen while the CPU is in debug mode.

Bit 4 **GPDMA_4**: GPDMA channel 4 stop in debug
0: normal operation. GPDMA channel 4 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 4 is frozen while the CPU is in debug mode.

Bit 3 **GPDMA_3**: GPDMA channel 3 stop in debug
0: normal operation. GPDMA channel 3 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 3 is frozen while the CPU is in debug mode.

Bit 2 **GPDMA_2**: GPDMA channel 2 stop in debug
0: normal operation. GPDMA channel 2 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 2 is frozen while the CPU is in debug mode.

Bit 1 **GPDMA_1**: GPDMA channel 1 stop in debug
0: normal operation. GPDMA channel 1 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 1 is frozen while the CPU is in debug mode.

Bit 0 **GPDMA_0**: GPDMA channel 0 stop in debug
0: normal operation. GPDMA channel 0 continues to operate while the CPU is in debug mode.
1: stop in debug. GPDMA channel 0 is frozen while the CPU is in debug mode.

**DBGMCU APB1 peripheral freeze register (DBGMCU_APB1FZR)**

Address offset: 0x03C
Reset value: 0x0000 0000
Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **I2C3**: I2C3 SMBUS timeout stop in debug
0: Normal operation - I2C3 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I2C3 SMBUS timeout is frozen while Cortex-M7 is in Debug mode

Bit 22  **I2C2**: I2C2 SMBUS timeout stop in debug
0: Normal operation - I2C2 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I2C2 SMBUS timeout is frozen while Cortex-M7 is in Debug mode

Bit 21  **I2C1**: I2C1 SMBUS timeout stop in debug
0: Normal operation - I2C1 SMBUS timeout continues operating while the core is in Debug mode
1: Stop in debug - I2C1 SMBUS timeout is frozen while the core is in Debug mode

Bits 20:12  Reserved, must be kept at reset value.

Bit 11  **WWDG**: WWDG stop in debug
0: Normal operation - WWDG continues to operate while the Cortex-M7 is in Debug mode
1: Stop in debug - WWDG is frozen while the Cortex-M7 is in Debug mode

Bit 10  Reserved, must be kept at reset value.

Bit 9  **LPTIM1**: LPTIM1 stop in debug
0: Normal operation - LPTIM1 continues operating while the core is in Debug mode
1: Stop in debug - LPTIM1 is frozen while Cortex-M7 is in Debug mode

Bit 8  **TIM14**: TIM14 stop in debug
0: Normal operation - TIM14 continues operating while the core is in Debug mode
1: Stop in debug - TIM14 is frozen while Cortex-M7 is in Debug mode

Bit 7  **TIM13**: TIM13 stop in debug
0: Normal operation - TIM13 continues operating while the core is in Debug mode
1: Stop in debug - TIM13 is frozen while Cortex-M7 is in Debug mode

Bit 6  **TIM12**: TIM12 stop in debug
0: Normal operation - TIM12 continues operating while the core is in Debug mode
1: Stop in debug - TIM12 is frozen while Cortex-M7 is in Debug mode

Bit 5  **TIM7**: TIM7 stop in debug
0: Normal operation - TIM7 continues operating while the core is in Debug mode
1: Stop in debug - TIM7 is frozen while Cortex-M7 is in Debug mode

Bit 4  **TIM6**: TIM6 stop in debug
0: Normal operation - TIM6 continues operating while the core is in Debug mode
1: Stop in debug - TIM6 is frozen while Cortex-M7 is in Debug mode

Bit 3  **TIM5**: TIM5 stop in debug
0: Normal operation - TIM5 continues operating while the core is in Debug mode
1: Stop in debug - TIM5 is frozen while Cortex-M7 is in Debug mode
Bit 2 **TIM4**: TIM4 stop in debug
0: Normal operation - TIM4 continues operating while the core is in Debug mode
1: Stop in debug - TIM4 is frozen while Cortex-M7 is in Debug mode

Bit 1 **TIM3**: TIM3 stop in debug
0: Normal operation - TIM3 continues operating while the core is in Debug mode
1: Stop in debug - TIM3 is frozen while Cortex-M7 is in Debug mode

Bit 0 **TIM2**: TIM2 stop in debug
0: Normal operation - TIM2 continues operating while the core is in Debug mode
1: Stop in debug - TIM2 is frozen while Cortex-M7 is in Debug mode

**DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZ)**
Address offset: 0x04C
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TIM9**: TIM9 stop in debug
0: Normal operation - TIM9 continues operating while the core is in Debug mode
1: Stop in debug - TIM9 is frozen and TIM9 outputs are disabled while Cortex-M7 is in Debug mode

Bit 18 **TIM17**: TIM17 stop in debug
0: Normal operation - TIM17 continues operating while the core is in Debug mode
1: Stop in debug - TIM17 is frozen and TIM17 outputs are disabled while Cortex-M7 is in Debug mode

Bit 17 **TIM16**: TIM16 stop in debug
0: Normal operation - TIM16 continues operating while the core is in Debug mode
1: Stop in debug - TIM16 is frozen and TIM16 outputs are disabled while Cortex-M7 is in Debug mode

Bit 16 **TIM15**: TIM15 stop in debug
0: Normal operation - TIM15 continues operating while the core is in Debug mode
1: Stop in debug - TIM15 is frozen and TIM15 outputs are disabled while Cortex-M7 is in Debug mode

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **TIM1**: TIM1 stop in debug
0: Normal operation - TIM1 continues operating while the core is in Debug mode
1: Stop in debug - TIM1 is frozen and TIM1 outputs are disabled while Cortex-M7 is in Debug mode.

**DBGMCU APB4 peripheral freeze register (DBGMCU_APB4FZR)**
Address offset: 0x054
### DBGMCU status register (DBGMCU_SR)

**Address offset:** 0x0FC

**Reset value:** 0x0001 0003

**Description:** This register is always accessible.

#### Bits 31:19
Reserved, must be kept at reset value.

#### Bit 18 **IWDG**
- Independent watchdog for stop in debug
- 0: Normal operation - watchdog continues counting while the core is in Debug mode
- 1: Stop in debug - watchdog is frozen while Cortex-M7 is in Debug mode

#### Bit 17 Reserved, must be kept at reset value.

#### Bit 16 **RTC**
- RTC stop in debug
- 0: Normal operation - RTC continues operating while the core is in Debug mode
- 1: Stop in debug - RTC is frozen while Cortex-M7 is in Debug mode

#### Bits 15:13
Reserved, must be kept at reset value.

#### Bit 12 **LPTIM5**
- LPTIM5 stop in debug
- 0: Normal operation - LPTIM5 continues operating while the core is in Debug mode
- 1: Stop in debug - LPTIM5 is frozen while Cortex-M7 is in Debug mode

#### Bit 11 **LPTIM4**
- LPTIM4 stop in debug
- 0: Normal operation - LPTIM4 continues operating while the core is in Debug mode
- 1: Stop in debug - LPTIM4 is frozen while Cortex-M7 is in Debug mode

#### Bit 10 **LPTIM3**
- LPTIM2 stop in debug
- 0: Normal operation - LPTIM2 continues operating while the core is in Debug mode
- 1: Stop in debug - LPTIM2 is frozen while Cortex-M7 is in Debug mode

#### Bit 9 **LPTIM2**
- LPTIM2 stop in debug
- 0: Normal operation - LPTIM2 continues operating while the core is in Debug mode
- 1: Stop in debug - LPTIM2 is frozen while Cortex-M7 is in Debug mode

#### Bits 8:0
Reserved, must be kept at reset value.

---

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### AP_ENABLED[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

#### AP_PRESENT[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:16 **AP_ENABLED[15:0]**: Bit n identifies whether access port AP n is open (can be accessed via the debug port) or locked (debug access to the AP is blocked)
- Bit n = 0: APn locked
- Bit n = 1: APn enabled

Bits 15:0 **AP_PRESENT[15:0]**: Bit n identifies whether access port AP n is present in the device
- Bit n = 0: APn absent
- Bit n = 1: APn present

### DBGMCU debug authentication mailbox host register
(DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100
Reset value: 0xXXXX XXXX
Description: This register is read only when accessed by the CPU, writes have no effect. It register can be written and read by an external debugger.

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

Bits 31:0 **MESSAGE[31:0]**: Debug host to device mailbox message.
During debug authentication the debug host communicates with the device via this register.

### DBGMCU debug authentication mailbox device register
(DBGMCU_DBG_AUTH_DEVICE)

Address offset: 0x104
Reset value: 0xXXXX XXXX
Description: This register is read only when accessed via the debug port, writes have no effect. This register can be written and read by the CPU.

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

Bits 31:0 **MESSAGE[31:0]**: Device to debug host mailbox message.
During debug authentication the device communicates with the debug host via this register.
**DBGMCU debug authentication mailbox acknowledge register (DBGMCU_DBG_AUTH_ACK)**

Address offset: 0x108  
Reset value: 0x0000 0000  

Description: This register is always accessible.

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

- **Bit 1 DEV_ACK**: Device to device acknowledge. The host sets this bit to indicate that it has placed a message in the DBGMCU_DBG_AUTH_HOST register. It is reset by the device after reading the message.

- **Bit 0 HOST_ACK**: Host to device acknowledge. The device sets this bit to indicate that it has placed a message in the DBGMCU_DBG_AUTH_DEVICE register. It should be reset by the host after reading the message.

**DBGMCU peripheral identity register 4 (DBGMCU_PIDR4)**

Address offset: 0xFD0  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIZE[3:0]</td>
<td>JEP106CON[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- **Bits 7:4 SIZE[3:0]**: Register file size  
  0x0: Register file occupies a single 4 Kbyte region

- **Bits 3:0 JEP106CON[3:0]**: JEP106 continuation code  
  0x0: STMicroelectronics JEDEC code
### DBGMCU peripheral identity register 0 (DBGMCU_PIDR0)

**Address offset:** 0xFE0  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PARTNUM[7:0]:** Part number field, bits [7:0]  
0x0: DBGMCU

### DBGMCU peripheral identity register 1 (DBGMCU_PIDR1)

**Address offset:** 0xFE4  
**Reset value:** 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **JEP106ID[3:0]:** JEP106 identity code field, bits [3:0]  
0x0: STMicroelectronics JEDEC code

Bits 3:0  **PARTNUM[11:8]:** Part number field, bits [11:8]  
0x0: DBGMCU

### DBGMCU peripheral identity register 2 (DBGMCU_PIDR2)

**Address offset:** 0xFE8  
**Reset value:** 0x0000 0000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]:** Revision field, bits [3:0]  
0x0: STMicroelectronics JEDEC code

Bits 3:0  **JEDEC JEP106ID[6:4]:** JEDEC identification field, bits [6:4]  
0x0: STMicroelectronics JEDEC code
DBGMCU peripheral identity register 3 (DBGMCU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4 **REVAND[3:0]**: Metal fix version

0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified

0x0: No customer modifications

DBGMCU component identity register (DBGMCU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]

0x0D: Common ID value
### DBGMCU component identity register (DBGMCU_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 00F0

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
- 0xF: System component with non-standard register layout

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
- 0x0: Common ID value

### DBGMCU component identity register (DBGMCU_CIDR2)

Address offset: 0xFF8  
Reset value: 0x0000 0005

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
- 0x05: Common ID value

### DBGMCU component identity register (DBGMCU_CIDR3)

Address offset: 0xFFC  
Reset value: 0x0000 00B1

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>30</td>
<td>29</td>
<td>28</td>
<td>27</td>
<td>26</td>
<td>25</td>
<td>24</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>20</td>
<td>19</td>
<td>18</td>
<td>17</td>
<td>16</td>
</tr>
</tbody>
</table>

<p>| | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [23:16]
- 0x05: Common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]

0xB1: Common ID value
### 66.8.2 DBGMCU register map and reset values

Table 784. DBGMCU register map and reset values

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>DBGCARD_IDCODE</th>
<th>REV_ID[15:0]</th>
<th>DEV_ID[11:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x000</td>
<td>DBGCARD_IDCODE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x40</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x04</td>
<td>DBGCARD_CR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0x08</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0C</td>
<td>DBGCARD_AHB5FZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x20</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x24</td>
<td>DBGCARD_AHB1FZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x28</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2C</td>
<td>DBGCARD_APB1FZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x40</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x44</td>
<td>DBGCARD_APB1HF</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x4C</td>
<td>DBGCARD_APB2FZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x54</td>
<td>DBGCARD_APB4FZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x58</td>
<td></td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC</td>
<td>DBGCARD_SR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reset value</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0x100</td>
<td>DBGCARD_AUTH_H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESSAGE[31:0]</td>
<td>XXXXXXXXXX</td>
<td>XXXXXXXXXX</td>
</tr>
<tr>
<td>0x104</td>
<td>DBGCARD_AUTH_D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MESSAGE[31:0]</td>
<td>XXXXXXXXXX</td>
<td>XXXXXXXXXX</td>
</tr>
</tbody>
</table>
### Table 784. DBGMCU register map and reset values (continued)

| Offset | Register name       | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|---------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x108  | DBGMCU_AUTH_ACK     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x10C to 0xFFFF | Reserved           |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD0  | DBGMCU_PIDR4        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFE0  | DBGMCU_PIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFE4  | DBGMCU_PIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFE8  | DBGMCU_PIDR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  |
| 0xFEC  | DBGMCU_PIDR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0xFF0  | DBGMCU_CIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 0  |
| 0xFF4  | DBGMCU_CIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value         | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 0  | 0  |

Reset values for various registers are provided, indicating the initial state of each bit field upon power-up or reset.
### Table 784. DBGMCU register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFF8</td>
<td>DBGMCU_CIDR2</td>
<td>00000000000000000000000000000101</td>
</tr>
<tr>
<td>0xFFC</td>
<td>DBGMCU_CIDR3</td>
<td>00000000000000000000000010110001</td>
</tr>
</tbody>
</table>
66.9  **Cortex-M7 debug functional description**

The Cortex-M7 subsystem features the following CoreSight™ components:

- ROM tables
- System control space (SCS)
- Breakpoint unit (FPB)
- Data watchpoint and trace unit (DWT)
- Instrumentation trace macrocell (ITM)
- Embedded trace macrocell (ETM)
- Cross trigger interface (CTI)

These components are accessible by the debugger via the Cortex-M7 AHB-AP and its associated AHBD bus.

66.9.1  **Cortex-M7 ROM tables**

The ROM table is a CoreSight™ component that contains the base addresses of all the CoreSight debug components accessible via the AHBD. These tables allow a debugger to discover the topology of the CoreSight system automatically.

There are two ROM tables in the Cortex-M7 sub-system:

- Cortex-M7 CPU ROM table
  - This table is pointed to by the BASE register in the Cortex-M7 AHB-AP. It contains the base address pointers for the ETM and CTI, as well as for the Cortex-M7 PPB ROM table.

- Cortex-M7 PPB (private peripheral bus) ROM table
  - This table contains pointers to the Cortex-M7 System Control Space registers allowing the debugger to identify the CPU core, as well as to the remaining CoreSight components in the Cortex-M7 subsystem: FPB, DWT and ITM.

The CPU ROM table occupies a 4-Kbyte, 32-bit wide chunk of AHBD address space, from 0xE00FE000 to 0xE00FEFFC.

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FE000</td>
<td>Cortex-M7 PPB  ROM table</td>
<td>0xE00FF000</td>
<td>0x00001000</td>
<td>4 Kbyte</td>
<td>0x00001003</td>
</tr>
<tr>
<td>0xE00FE004</td>
<td>Cortex-M7 ETM</td>
<td>0xE0041000</td>
<td>0xFFF43000</td>
<td>4 Kbyte</td>
<td>0xFFF43003</td>
</tr>
<tr>
<td>0xE00FE008</td>
<td>Cortex-M7 CTI</td>
<td>0xE0043000</td>
<td>0xFFF45000</td>
<td>4 Kbyte</td>
<td>0xFFF45003</td>
</tr>
<tr>
<td>0xE00FE00C</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x1FF02002</td>
</tr>
<tr>
<td>0xE00FE010</td>
<td>Top of table</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FE010 to 0xE00FEFC8</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FEFCC to 0xE00FEFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 787</td>
</tr>
</tbody>
</table>

Table 785. Cortex-M7 CPU ROM table
The Cortex-M7 PPB ROM table occupies a 4-Kbyte, 32-bit wide chunk of APB-D address space, from 0xE00FF000 to 0xE00FFFFC.

Table 786. Cortex-M7 PPB ROM table

<table>
<thead>
<tr>
<th>Address in ROM table</th>
<th>Component name</th>
<th>Component base address</th>
<th>Component address offset</th>
<th>Size</th>
<th>Entry</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE00FF000</td>
<td>SCS</td>
<td>0xE000E000</td>
<td>0xFFF0F000</td>
<td>4 Kbyte</td>
<td>0xFFF0F003</td>
</tr>
<tr>
<td>0xE00FF004</td>
<td>DWT</td>
<td>0xE0001000</td>
<td>0xFFF02000</td>
<td>4 Kbyte</td>
<td>0xFFF02003</td>
</tr>
<tr>
<td>0xE00FF008</td>
<td>FPB</td>
<td>0xE0002000</td>
<td>0xFFF03000</td>
<td>4 Kbyte</td>
<td>0xFFF03003</td>
</tr>
<tr>
<td>0xE00FF00C</td>
<td>ITM</td>
<td>0xE0000000</td>
<td>0xFFF01000</td>
<td>4 Kbyte</td>
<td>0xFFF01003</td>
</tr>
<tr>
<td>0xE00FF010</td>
<td>TPIU(1)</td>
<td>0xE0040000</td>
<td>0xFFF41000</td>
<td>4 Kbyte</td>
<td>0xFFF41002</td>
</tr>
<tr>
<td>0xE00FF014</td>
<td>ETM(1)</td>
<td>0xE0041000</td>
<td>0xFFF42000</td>
<td>4 Kbyte</td>
<td>0xFFF42002</td>
</tr>
<tr>
<td>0xE00FF018</td>
<td>Top of table</td>
<td></td>
<td></td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FF01C to 0xE00FFFFC</td>
<td>Reserved</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0x00000000</td>
</tr>
<tr>
<td>0xE00FFFFCC to 0xE00FFFFC</td>
<td>ROM table registers</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>See Table 788</td>
</tr>
</tbody>
</table>

1. The TPIU and ETM are included in this table by default, but bit 0 is reset to indicate that they are not present.

The Topology for the CoreSight™ components in the Cortex-M7 subsystem is shown in Figure 1009.

Figure 1009. Cortex-M7 CoreSight Topology
66.9.2 Cortex-M7 CPU ROM registers

CPU ROM memory type register (M7_CPUROM_MEMTYPE)

Address offset: 0xFCC
Reset value: 0x0000 0001

| Bits 31:1 | Reserved, must be kept at reset value. |
| Bits 0  | SYSMEM: System memory presence |
| 1: System memory is present on this bus |

CPU ROM CoreSight peripheral identity register 4 (M7_CPUROM_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0000

| Bits 31:8 | Reserved, must be kept at reset value. |
| Bits 7:4 | SIZE[3:0]: Register file size |
| 0x0: Register file occupies a single 4 Kbyte region |
| Bits 3:0 | JEP106CON[3:0]: JEP106 continuation code |
| 0x0: STMicroelectronics JEDEC continuation code |

CPU ROM CoreSight peripheral identity register 0 (M7_CPUROM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0085

| Bits 31:16 | Reserved, must be kept at reset value. |
| Bits 15:8 | PARTNUM[7:0] |
| | STMicroelectronics part number |
| | STMicroelectronics part number |

| Bits 7:4 | SIZE[3:0] |
| 0x0: Register file occupies a single 4 Kbyte region |
| Bits 3:0 | JEP106CON[3:0]: JEP106 continuation code |
| 0x0: STMicroelectronics JEDEC continuation code |
CPU ROM CoreSight peripheral identity register 1 (M7_CPUROM_PIDR1)

Address offset: 0xFE4
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]
0x85: STM32H7Rx/7Sx

CPU ROM CoreSight peripheral identity register 2 (M7_CPUROM_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: STMicroelectronics

Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: STM32H7Rx/7Sx
### CPU ROM CoreSight peripheral identity register 3 (M7_CPUROM_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

| Bits 7:4 | REVAND[3:0]: Metal fix version  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0: No metal fix</td>
<td></td>
</tr>
</tbody>
</table>

| Bits 3:0 | CMOD[3:0]: Customer modified  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0: No customer modifications</td>
<td></td>
</tr>
</tbody>
</table>

### CPU ROM CoreSight component identity register 0 (M7_CPUROM_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

| Bits 7:0 | PREAMBLE[7:0]: Component ID field, bits [7:0]  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0D: Common ID value</td>
<td></td>
</tr>
</tbody>
</table>

### CPU ROM CoreSight component identity register 1 (M7_CPUROM_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 7:0</th>
<th>PREAMBLE[11:8]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0D: Common ID value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

**CPU ROM CoreSight component identity register 2 (M7_CPUROM_CIDR2)**
Address offset: 0xFF8
Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[19:12]</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

**CPU ROM CoreSight component identity register 3 (M7_CPUROM_CIDR3)**
Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PREAMBLE[27:20]</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value
### Cortex-M7 CPU ROM table register map and reset values

**Table 787. Cortex-M7 CPU ROM table register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
<th>Offset</th>
<th>Register name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>M7_CPUROM_MEMTYPE</td>
<td>0xFD0</td>
<td>M7_CPUROM_PIDR4</td>
<td>0xFD4 to 0xFD0</td>
<td>Reserved</td>
<td>0xFE0</td>
<td>M7_CPUROM_PIDR0</td>
<td>0xFE4</td>
<td>M7_CPUROM_PIDR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td></td>
<td>- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -</td>
<td>0xFEC</td>
<td>M7_CPUROM_PIDR3</td>
<td>0xFF0</td>
<td>M7_CPUROM_CIDR0</td>
<td>0xFF4</td>
<td>M7_CPUROM_CIDR1</td>
<td>0xFF8</td>
<td>M7_CPUROM_CIDR2</td>
</tr>
<tr>
<td>0xFD0</td>
<td>M7_CPUROM_PIDR4</td>
<td>0xFE0</td>
<td>M7_CPUROM_PIDR0</td>
<td>0xFE4</td>
<td>M7_CPUROM_PIDR1</td>
<td>0xFE8</td>
<td>M7_CPUROM_PIDR2</td>
<td>0xFEC</td>
<td>M7_CPUROM_PIDR3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xFD4</td>
<td>Reserved</td>
<td>0xFE0</td>
<td>M7_CPUROM_PIDR0</td>
<td>0xFE4</td>
<td>M7_CPUROM_PIDR1</td>
<td>0xFE8</td>
<td>M7_CPUROM_PIDR2</td>
<td>0xFEC</td>
<td>M7_CPUROM_PIDR3</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_CPUROM_PIDR0</td>
<td>0xFE4</td>
<td>M7_CPUROM_PIDR1</td>
<td>0xFE8</td>
<td>M7_CPUROM_PIDR2</td>
<td>0xFF0</td>
<td>M7_CPUROM_CIDR0</td>
<td>0xFF4</td>
<td>M7_CPUROM_CIDR1</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_CPUROM_PIDR2</td>
<td>0xFF0</td>
<td>M7_CPUROM_CIDR0</td>
<td>0xFF4</td>
<td>M7_CPUROM_CIDR1</td>
<td>0xFF8</td>
<td>M7_CPUROM_CIDR2</td>
<td>0xFFC</td>
<td>M7_CPUROM_CIDR3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
<tr>
<td>0xFF0</td>
<td>M7_CPUROM_CIDR0</td>
<td>0xFF4</td>
<td>M7_CPUROM_CIDR1</td>
<td>0xFF8</td>
<td>M7_CPUROM_CIDR2</td>
<td>0xFFC</td>
<td>M7_CPUROM_CIDR3</td>
<td>0xFFC</td>
<td>M7_CPUROM_CIDR3</td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
<td></td>
<td>Reset value</td>
</tr>
</tbody>
</table>

**Reset values**

- **0xFD0**: M7_CPUROM_PIDR4
  - Reset value
  - Offset: 27-0
  - CEL: 0xFD0
  - JEP106ID[3:0]: 0101
  - JEP106ID[6:4]: 0001
- **0xFE0**: M7_CPUROM_PIDR0
  - Reset value
  - Offset: 27-0
  - PARTNUM[7:0]: 10101
- **0xFE4**: M7_CPUROM_PIDR1
  - Reset value
  - Offset: 27-0
  - PARTNUM[1:8]: 0100
- **0xFE8**: M7_CPUROM_PIDR2
  - Reset value
  - Offset: 27-0
  - REVAND[3:0]: 0000
  - CMOD[3:0]: 0000
- **0xFF0**: M7_CPUROM_CIDR0
  - Reset value
  - Offset: 19-0
  - PREAMBLE[7:0]: 0110
- **0xFF4**: M7_CPUROM_CIDR1
  - Reset value
  - Offset: 19-0
  - CLASS[3:0]: 0000
  - PREAMBLE[11:8]: 0100
- **0xFF8**: M7_CPUROM_CIDR2
  - Reset value
  - Offset: 19-0
  - Reserved: 0000
  - PREAMBLE[19:12]: 0101
- **0xFFC**: M7_CPUROM_CIDR3
  - Reset value
  - Offset: 19-0
  - Reserved: 0000
  - PREAMBLE[27:20]: 0101

**Register names**

- M7_CPUROM_MEMTYPE
- M7_CPUROM_PIDR4
- M7_CPUROM_PIDR0
- M7_CPUROM_PIDR1
- M7_CPUROM_PIDR2
- M7_CPUROM_PIDR3
- M7_CPUROM_CIDR0
- M7_CPUROM_CIDR1
- M7_CPUROM_CIDR2
- M7_CPUROM_CIDR3
66.9.4 Cortex-M7 PPB ROM registers

**PPB ROM memory type register (M7_PPBROM_MEMTYPE)**

Address offset: 0xFCC
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus

**PPB ROM CoreSight peripheral identity register 4 (M7_PPBROM_PIDR4)**

Address offset: 0xFD0
Reset value: 0x0000 0004

| Bit 31 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| 15     | 14     | 13     | 12     | 11     | 10     | 9      | 8      | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |

| Bit 31:8 Reserved, must be kept at reset value. |
| Bit 7:4 **SIZE[3:0]**: Register file size |
0x0: Register file occupies a single 4 Kbyte region

| Bit 3:0 **JEP106CON[3:0]**: JEP106 continuation code |
0x4: Arm® JEDEC continuation code

**PPB ROM CoreSight peripheral identity register 0 (M7_PPBROM_PIDR0)**

Address offset: 0xFE0
Reset value: 0x0000 00C7
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]
0x3: Cortex-M7 PPB ROM table

**PPB ROM CoreSight peripheral identity register 1 (M7_PPBROM_PIDR1)**

Address offset: 0xFE4
Reset value: 0x0000 00B4

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x3: Arm® JEDEC code

Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: Cortex-M7 PPB ROM table

**PPB ROM CoreSight peripheral identity register 2 (M7_PPBROM_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 00B4

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3  **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code
### PPB ROM CoreSight peripheral identity register 3 (M7_PPBROM_PIDR3)

Address offset: 0xFEC  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-16</th>
<th>Bit 15-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVAND[3:0]**: Metal fix version
- 0x0: No metal fix

Bits 3:0  **CMOD[3:0]**: Customer modified
- 0x0: No customer modifications

### PPB ROM CoreSight component identity register 0 (M7_PPBROM_CIDR0)

Address offset: 0xFF0  
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-16</th>
<th>Bit 15-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[7:0]**: Component ID field, bits [7:0]
- 0x0D: Common ID value

### PPB ROM CoreSight component identity register 1 (M7_PPBROM_CIDR1)

Address offset: 0xFF4  
Reset value: 0x0000 0010

<table>
<thead>
<tr>
<th>Bit 31-24</th>
<th>Bit 23-16</th>
<th>Bit 15-8</th>
<th>Bit 7-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[11:8]**: Component ID field, bits [7:0]
- 0x0D: Common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class
         0x1: ROM table component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]
         0x0: Common ID value

**PPB ROM CoreSight component identity register 2 (M7_PPBROM_CIDR2)**

Address offset: 0xFF8

Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]
         0x05: Common ID value

**PPB ROM CoreSight component identity register 3 (M7_PPBROM_CIDR3)**

Address offset: 0xFFC

Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[27:20]**: Component ID field, bits [31:24]
         0xB1: Common ID value
# Cortex-M7 PPB ROM table register map and reset values

Table 788. Cortex-M7 PPB ROM table register map and reset values

| Offset | Register name          | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|------------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0xFCC  | M7_PPBROM_MEMTYPE      |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1  |    |    |    |    |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFD0  | M7_PPBROM_PIDR4        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 1  | 0  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFD4  | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reserved               |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE0  | M7_PPBROM_PIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1  | 0  | 0  | 0  | 1  | 1  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFE4  | M7_PPBROM_PIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1  | 0  | 1  | 1  | 0  | 1  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFE8  | M7_PPBROM_PIDR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 1  | 0  | 1  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFEC  | M7_PPBROM_PIDR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 0  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFF0  | M7_PPBROM_CIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 1  | 1  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFF4  | M7_PPBROM_CIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 1  | 0  | 0  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFF8  | M7_PPBROM_CIDR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 0  | 0  | 0  | 0  | 0  | 1  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFFC  | M7_PPBROM_CIDR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    | 1  | 0  | 1  | 0  | 0  | 0  |
|        | Reset value            | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
66.9.6 Cortex-M7 data watchpoint and trace unit (DWT)

The DWT provides four comparators that can be used as:
- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (comparator 1 only)
- clock cycle counter comparator (comparator 0 only)

It also contains counters for:
- clock cycles
- folded instructions
- load store unit (LSU) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares one of the following with the value held in its DWT_COMP register:
- a data address
- an instruction address
- a data value
- the cycle count value, for comparator 0 only.

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:
- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value
- a watchpoint debug event, on either the PC value or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter Debug state.

For more details on how to use the DWT, refer to the Arm®v7-M Architecture Reference Manual [5].
66.9.7 Cortex-M7 DWT registers

DWT control register (M7_DWT_CTRL)

Address offset: 0x0000
Reset value: 0x0000

<table>
<thead>
<tr>
<th>Bits 31:28</th>
<th>NUMCOMP[3:0]: Number of implemented comparators (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 27</td>
<td>NOTRCPKT: Trace sampling and exception tracing support (read-only)</td>
</tr>
<tr>
<td></td>
<td>0: Supported</td>
</tr>
<tr>
<td>Bit 26</td>
<td>NOEXTRIG: External match signal, CMPMATCH support (read-only)</td>
</tr>
<tr>
<td></td>
<td>0: Supported</td>
</tr>
<tr>
<td>Bit 25</td>
<td>NOCYCCNT: Cycle counter support (read-only)</td>
</tr>
<tr>
<td></td>
<td>0: Supported</td>
</tr>
<tr>
<td>Bit 24</td>
<td>NOPRFCNT: Profiling counter support (read-only)</td>
</tr>
<tr>
<td></td>
<td>0: Supported</td>
</tr>
<tr>
<td>Bit 23</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 22</td>
<td>CYCEVTENA: Enable POSTCNT underflow event counter packet generation</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bit 21</td>
<td>FOLDEVTENA: Enable folded instruction counter overflow event generation</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bit 20</td>
<td>LSUEVTENA: Enable LSU counter overflow event generation</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bit 19</td>
<td>SLEEP EVTENA: Enable sleep counter overflow event generation</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bit 18</td>
<td>EXCEVTENA: Enable exception overhead counter overflow event generation</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bit 17</td>
<td>CPIEVTENA: Enable CPI counter overflow event generation</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>
Bit 16 **EXCTRKENA**: Enable exception trace generation

0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: POSTCNT counter use enable

Enables the use of POSTCNT counter as a timer for Periodic PC sample packet generation.

0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Position of synchronization packet counter tap on CYCCNT counter

This selection determines the synchronization packet rate.

0x0: Disabled - no synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Position of the POSTCNT tap on the CYCCNT counter

0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter

Writes to this field are ignored if POSTCNT counter is enabled (that is, CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTRESET[3:0]**: Reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: CYCCNT counter enable

0: Disabled
1: Enabled

**DWT cycle count register (M7_DWT_CYCCNT)**

Address offset: 0x004

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th></th>
<th>CYCCNT[31:16]</th>
<th></th>
<th>CYCCNT[15:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>rw</td>
<td>30</td>
<td>rw</td>
</tr>
<tr>
<td>29</td>
<td>rw</td>
<td>28</td>
<td>rw</td>
</tr>
<tr>
<td>27</td>
<td>rw</td>
<td>26</td>
<td>rw</td>
</tr>
<tr>
<td>25</td>
<td>rw</td>
<td>24</td>
<td>rw</td>
</tr>
<tr>
<td>23</td>
<td>rw</td>
<td>22</td>
<td>rw</td>
</tr>
<tr>
<td>21</td>
<td>rw</td>
<td>20</td>
<td>rw</td>
</tr>
<tr>
<td>19</td>
<td>rw</td>
<td>18</td>
<td>rw</td>
</tr>
<tr>
<td>17</td>
<td>rw</td>
<td>16</td>
<td>rw</td>
</tr>
<tr>
<td>15</td>
<td>rw</td>
<td>14</td>
<td>rw</td>
</tr>
<tr>
<td>13</td>
<td>rw</td>
<td>12</td>
<td>rw</td>
</tr>
<tr>
<td>11</td>
<td>rw</td>
<td>10</td>
<td>rw</td>
</tr>
<tr>
<td>9</td>
<td>rw</td>
<td>8</td>
<td>rw</td>
</tr>
<tr>
<td>7</td>
<td>rw</td>
<td>6</td>
<td>rw</td>
</tr>
<tr>
<td>5</td>
<td>rw</td>
<td>4</td>
<td>rw</td>
</tr>
<tr>
<td>3</td>
<td>rw</td>
<td>2</td>
<td>rw</td>
</tr>
<tr>
<td>1</td>
<td>rw</td>
<td>0</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **CYCCNT[31:0]**: Processor clock cycle counter
### DWT CPI count register (M7_DWT_CPICNT)

Address offset: 0x008  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **CPICNT[7:0]**: CPI counter  
Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.

### DWT exception count register (M7_DWT_EXCCNT)

Address offset: 0x00C  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter  
Counts the number of cycles spent in exception processing.

### DWT sleep count register (M7_DWT_SLPCNT)

Address offset: 0x010  
Reset value: 0x0000 0000  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:8 Reserved, must be kept at reset value.  
Bits 7:0 **SLEEPCNT[7:0]**: Exception overhead cycle counter  
Counts the number of cycles spent in exception processing.
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  SLEEP_CNT[7:0]: Sleep cycle counter
Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

**DWT LSU count register (M7_DWT_LSUCNT)**

Address offset: 0x014
Reset value: 0x0000 0000

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

Bits 7:0  LSUCNT[7:0]: Load store counter
Counts additional cycles required to execute load and store instructions.

**DWT fold count register (M7_DWT_FOLDCNT)**

Address offset: 0x018
Reset value: 0x0000 0000

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

Bits 7:0  FOLDCNT[7:0]: Folded instruction counter
Increments on each instruction that takes 0 cycles.

**DWT program counter sample register (M7_DWT_PCSR)**

Address offset: 0x01C
Reset value: 0x0000 0000

**EIASAMPLE[31:16]**

**EIASAMPLE[15:0]**
Bits 31:0  **EIASAMPLE[31:0]**: Executed instruction address sample value  
Samples the current value of the program counter.

**DWT comparator register x (M7_DWT_COMPx)**

Address offset: 0x020 + x * 0x10 (x = 0 to 3)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>EIASAMPLE[31:0]</th>
<th>Executed instruction address sample value</th>
<th>Samples the current value of the program counter.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x020 + x * 0x10 (x = 0 to 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP[31:16]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMP[15:0]</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **COMP[31:0]**: Reference value for comparison.

**DWT mask register x (M7_DWT_MASKx)**

Address offset: 0x024 + x * 0x10 (x = 0 to 3)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:0</th>
<th>COMP[31:0]</th>
<th>Reference value for comparison.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Address offset</th>
<th>0x024 + x * 0x10 (x = 0 to 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset value</td>
<td>0x0000 0000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASK[4:0]</td>
<td>rw rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 4:0  **MASK[4:0]**: Comparator mask size  
Provides the size of the ignore mask applied to the access address for address range matching by comparator n. A debugger can write 0b11111 to this field and then read the register back to determine the maximum mask size supported.

**DWT function register x (M7_DWT_FUNCTx)**

Address offset: 0x028 + x * 0x10 (x = 0 to 3)  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATCHED</td>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAVADDR[3:0]</td>
<td>rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits</th>
<th>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATAVADDR[3:0]</td>
<td>rw rw rw rw</td>
</tr>
<tr>
<td>DATAVSIZE[1:0]</td>
<td>rw rw</td>
</tr>
<tr>
<td>LINK1E</td>
<td>NA</td>
</tr>
<tr>
<td>DATAV MATCH</td>
<td>MATCH</td>
</tr>
<tr>
<td>EMIT MATCH</td>
<td>CYC MATCH</td>
</tr>
<tr>
<td>EMIT RANG E</td>
<td>FUNCTION[3:0]</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
<td></td>
</tr>
</tbody>
</table>

3716/3791  RM0477 Rev 6
Bits 31:25  Reserved, must be kept at reset value.

Bit 24  MATCHED: Comparator match (read-only)
Indicates if a comparator match has occurred since the register was last read.

  0: No match
  1: Match occurred

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

Bits 19:16  DATAVADDR1[3:0]: Comparator number of a second comparator
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a second comparator to use for linked address comparison.

Bits 15:12  DATAVADDR0[3:0]: Comparator number of a comparator
When the DATAVMATCH and LNK1ENA bits are both 1, this field can hold the comparator number of a comparator to use for linked address comparison.

Bits 11:10  DATAVSIZE[1:0]: Size of required data comparison
For data value matching, specifies the size of the required data comparison.

  0x0: Byte
  0x1: Half word
  0x2: Word
  0x3: Reserved

Bit 9  LINK1ENA: Support of a second linked comparator (read-only)
Indicates whether the use of a second linked comparator is supported (read-only).

  1: Supported

Bit 8  DATAVMATCH: Cycle comparison enable

  0: Perform address comparison
  1: Perform data value comparison

Bit 7  CYCMATCH: Cycle count comparison enable on comparator 0
This field is reserved for other comparators.

  0: No cycle count comparison
  1: Compare DWT_COMP0 with the cycle counter, DWT_CYCCNT

Bit 6  Reserved, must be kept at reset value.

Bit 5  EMITRANGE: Data trace address offset packet enable

  Enables the generation of data trace address offset packets (containing data address bits 0 to 15)

  0: Disabled
  1: Enabled

Bit 4  Reserved, must be kept at reset value.

Bits 3:0  FUNCTION[3:0]: Action on comparator match
The meaning of this bit field depends on the setting of the DATAVMATCH and CYCMATCH fields. See [5].
### DWT CoreSight peripheral identity register 4 (M7_DWT_PIDR4)

Address offset: 0xFD0  
Reset value: 0x0000 0004  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>![](31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **SIZE[3:0]**: Register file size  
0x0: Register file occupies a single 4 Kbyte region

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

### DWT CoreSight peripheral identity register 0 (M7_DWT_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0002  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>![](31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]  
0x02: DWT part number

### DWT CoreSight peripheral identity register 1 (M7_DWT_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B0  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>![](31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **JEP106ID[3:0]**: JEP106 continuation code  
0x4: Arm® JEDEC code
Bits 31:8  Reserved, must be kept at reset value.

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

Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]
          0x0: DWT part number

### DWT CoreSight peripheral identity register 2 (M7_DWT_PIDR2)

Address offset: 0xFE8
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]**: Component revision number
          0x0: r0p0

Bit 3  **JEDEC**: JEDEC assigned value
       1: Designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
          0x3: Arm® JEDEC code

### DWT CoreSight peripheral identity register 3 (M7_DWT_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVAND[3:0]**: Metal fix version
          0x0: No metal fix

Bits 3:0  **CMOD[3:0]**: Customer modified
          0x0: No customer modifications
**DWT CoreSight component identity register 0 (M7_DWT_CIDR0)**

Address offset: 0xFF0  
Reset value: 0x0000 000D  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[7:0]**: Component ID field, bits [7:0]  
0x0D: Common ID value

**DWT CoreSight component identity register 1 (M7_DWT_CIDR1)**

Address offset: 0xFF4  
Reset value: 0x0000 00E0  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class  
0xE: Trace generator component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]  
0x0: Common ID value

**DWT CoreSight component identity register 2 (M7_DWT_CIDR2)**

Address offset: 0xFF8  
Reset value: 0x0000 0005  

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class  
0xE: Trace generator component

Bits 3:0  **PREAMBLE[19:12]**: Component ID field, bits [11:8]  
0x0: Common ID value
Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

### DWT CoreSight component identity register 3 (M7_DWT_CIDR3)

Address offset: 0xFFC
Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

### 66.9.8 Cortex-M7 DWT register map and reset values

The Cortex-M7 DWT registers are located at address range 0xE0001000 to 0xE0001FFC, on the AHBD.

#### Table 789. Cortex-M7 DWT register map and reset values

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 |  9 |  8 |  7 |  6 |  5 |  4 |  3 |  2 |  1 |  0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | M7_DWT_CTRL   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 1  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x004  | M7_DWT_CYCCNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x008  | M7_DWT_CPICNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0x00C  | M7_DWT_EXCCNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0x010  | M7_DWT_SLPCNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0x014  | M7_DWT_LSUCNT |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value   | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
Table 789. Cortex-M7 DWT register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Offset in register</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x018</td>
<td>M7_DWT_FOLDCNT</td>
<td></td>
<td>FOLDCNT[7:0]</td>
</tr>
<tr>
<td>0x01C</td>
<td>M7_DWT_PCSR</td>
<td></td>
<td>EIASAMPLE[31:0]</td>
</tr>
<tr>
<td>0x020</td>
<td>M7_DWT_COMP0</td>
<td></td>
<td>COMP[31:0]</td>
</tr>
<tr>
<td>0x024</td>
<td>M7_DWT_MASK0</td>
<td></td>
<td>MASK[4:0]</td>
</tr>
<tr>
<td>0x028</td>
<td>M7_DWT_FUNCT0</td>
<td></td>
<td>MATCHED, DATAVADDR[31:0], DATAVADDR[31:0], DATAVSIZE[1:0], LNK1ENA, DATAVMATCH, CYCMATCH, EMITRANGE, FUNCTION[3:0]</td>
</tr>
<tr>
<td>0x02C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>M7_DWT_COMP1</td>
<td></td>
<td>COMP[31:0]</td>
</tr>
<tr>
<td>0x034</td>
<td>M7_DWT_MASK1</td>
<td></td>
<td>MASK[4:0]</td>
</tr>
<tr>
<td>0x038</td>
<td>M7_DWT_FUNCT1</td>
<td></td>
<td>MATCHED, DATAVADDR[31:0], DATAVADDR[31:0], DATAVSIZE[1:0], LNK1ENA, DATAVMATCH, CYCMATCH, EMITRANGE, FUNCTION[3:0]</td>
</tr>
<tr>
<td>0x03C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x040</td>
<td>M7_DWT_COMP2</td>
<td></td>
<td>COMP[31:0]</td>
</tr>
<tr>
<td>0x044</td>
<td>M7_DWT_MASK2</td>
<td></td>
<td>MASK[4:0]</td>
</tr>
<tr>
<td>0x048</td>
<td>M7_DWT_FUNCT2</td>
<td></td>
<td>MATCHED, DATAVADDR[31:0], DATAVADDR[31:0], DATAVSIZE[1:0], LNK1ENA, DATAVMATCH, CYCMATCH, EMITRANGE, FUNCTION[3:0]</td>
</tr>
<tr>
<td>0x04C</td>
<td>Reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x050</td>
<td>M7_DWT_COMP3</td>
<td></td>
<td>COMP[31:0]</td>
</tr>
<tr>
<td>0x054</td>
<td>M7_DWT_MASK3</td>
<td></td>
<td>MASK[4:0]</td>
</tr>
</tbody>
</table>
## Table 789. Cortex-M7 DWT register map and reset values (continued)

| Offset | Register name   | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x058  | M7_DWT_FUNCT3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | 0  | -  | -  | -  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 0  |
| 0x05C to 0xFCC | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD0  | M7_DWT_PIDR4   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFD4 to 0xFCC | Reserved |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE0  | M7_DWT_PIDR0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFE4  | M7_DWT_PIDR1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFE8  | M7_DWT_PIDR2   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFE4  | M7_DWT_PIDR3   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFF0  | M7_DWT_CIDR0   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFF4  | M7_DWT_CIDR1   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFF8  | M7_DWT_CIDR2   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 0  | 0  |
| 0xFFC  | M7_DWT_CIDR3   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value    | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | 1  | 0  |

### 66.9.9 Cortex-M7 instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

1. **Software trace**
   
   Software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the identity of the port, the size of the write access, and the data written, into a packet that it writes to a FIFO. The ITM
Debug infrastructure

outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

2. Hardware trace
The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.

3. Local timestamping
The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset to zero every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.

4. Global system timestamping
Timesteps can also be generated using the system-wide 64-bit count value coming from the Timestamp Generator component.

66.9.10 Cortex-M7 ITM registers

ITM stimulus register x (M7_ITM_STIMx)
Address offset: 0x000 + x * 0x4 (x = 0 to 31)
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

Bits 31:0 **STIMULUS[31:0]**: Software event packet / FIFOREADY
Write data is output on the trace bus as a software event packet. When reading, bit 0 is a FIFOREADY indicator:
0: Stimulus port buffer is full (or port is disabled)
1: Stimulus port can accept new write data

ITM trace enable register (M7_ITM_TER)
Address offset: 0xE00
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>
Bits 31:0 **STIMENA[31:0]**: Stimulus port enable
   Each bit n (31:0) enables the stimulus port associated with the M7_ITM_STIMn register.
   0: Port disabled
   1: Port enabled

**ITM trace privilege registers (M7_ITM_TPR)**

Address offset: 0xE40
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports
   Each bit controls eight stimulus ports:
   0bXXX0: Unprivileged access permitted on ports 0 to 7
   0bXXX1: Only privileged access permitted on ports 0 to 7
   0bXX0X: Unprivileged access permitted on ports 8 to 15
   0bXX1X: Only privileged access permitted on ports 8 to 15
   0bX0XX: Unprivileged access permitted on ports 16 to 23
   0bX1XX: Only privileged access permitted on ports 16 to 23
   0b0XXX: Unprivileged access permitted on ports 24 to 31
   0b1XXX: Only privileged access permitted on ports 24 to 31

**ITM trace control register (M7_ITM_TCR)**

Address offset: 0xE80
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>GTSFREQ[1:0]</th>
<th>TSPRESCALE[1:0]</th>
<th>SWOE</th>
<th>TXEN</th>
<th>SYNC</th>
<th>TSEN</th>
<th>ITM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable
   Each bit n (31:0) enables the stimulus port associated with the M7_ITM_STIMn register.
   0: Port disabled
   1: Port enabled

**ITM trace privilege registers (M7_ITM_TPR)**

Address offset: 0xE40
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports
   Each bit controls eight stimulus ports:
   0bXXX0: Unprivileged access permitted on ports 0 to 7
   0bXXX1: Only privileged access permitted on ports 0 to 7
   0bXX0X: Unprivileged access permitted on ports 8 to 15
   0bXX1X: Only privileged access permitted on ports 8 to 15
   0bX0XX: Unprivileged access permitted on ports 16 to 23
   0bX1XX: Only privileged access permitted on ports 16 to 23
   0b0XXX: Unprivileged access permitted on ports 24 to 31
   0b1XXX: Only privileged access permitted on ports 24 to 31

**ITM trace control register (M7_ITM_TCR)**

Address offset: 0xE80
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>GTSFREQ[1:0]</th>
<th>TSPRESCALE[1:0]</th>
<th>SWOE</th>
<th>TXEN</th>
<th>SYNC</th>
<th>TSEN</th>
<th>ITM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable
   Each bit n (31:0) enables the stimulus port associated with the M7_ITM_STIMn register.
   0: Port disabled
   1: Port enabled

**ITM trace privilege registers (M7_ITM_TPR)**

Address offset: 0xE40
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports
   Each bit controls eight stimulus ports:
   0bXXX0: Unprivileged access permitted on ports 0 to 7
   0bXXX1: Only privileged access permitted on ports 0 to 7
   0bXX0X: Unprivileged access permitted on ports 8 to 15
   0bXX1X: Only privileged access permitted on ports 8 to 15
   0bX0XX: Unprivileged access permitted on ports 16 to 23
   0bX1XX: Only privileged access permitted on ports 16 to 23
   0b0XXX: Unprivileged access permitted on ports 24 to 31
   0b1XXX: Only privileged access permitted on ports 24 to 31

**ITM trace control register (M7_ITM_TCR)**

Address offset: 0xE80
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>GTSFREQ[1:0]</th>
<th>TSPRESCALE[1:0]</th>
<th>SWOE</th>
<th>TXEN</th>
<th>SYNC</th>
<th>TSEN</th>
<th>ITM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable
   Each bit n (31:0) enables the stimulus port associated with the M7_ITM_STIMn register.
   0: Port disabled
   1: Port enabled

**ITM trace privilege registers (M7_ITM_TPR)**

Address offset: 0xE40
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports
   Each bit controls eight stimulus ports:
   0bXXX0: Unprivileged access permitted on ports 0 to 7
   0bXXX1: Only privileged access permitted on ports 0 to 7
   0bXX0X: Unprivileged access permitted on ports 8 to 15
   0bXX1X: Only privileged access permitted on ports 8 to 15
   0bX0XX: Unprivileged access permitted on ports 16 to 23
   0bX1XX: Only privileged access permitted on ports 16 to 23
   0b0XXX: Unprivileged access permitted on ports 24 to 31
   0b1XXX: Only privileged access permitted on ports 24 to 31
Bits 31:24  Reserved, must be kept at reset value.

Bit 23  **BUSY**: ITM busy
Indicates whether the ITM is currently processing events (read-only):
0: Not busy
1: Busy

Bits 22:16  **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting
If multi-source trace is in use, the debugger must write a non-zero value to this field. Note: different IDs must be used for each trace source in the system.

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

Bits 11:10  **GTSFREQ[1:0]**: Global timestamp frequency
 Defines how often the ITM generates a global timestamp, based on the global timestamp clock frequency, or disables generation of global timestamps. The possible values are:
0x0: Disable generation of global timestamps
0x1: Generate timestamp request whenever the ITM detects a change in global timestamp counter bits [63:7]; this is approximately every 128 cycles
0x2: Generate timestamp request whenever the ITM detects a change in global timestamp counter bits [63:13]; this is approximately every 8192 cycles
0x3: Generate a timestamp after every packet, if the output FIFO is empty

Bits 9:8  **TSPRESCALE[1:0]**: Local timestamp prescale
Prescaler used with the trace packet reference clock The possible values are:
0x0: No prescaling
0x1: Divide by 4
0x2: Divide by 16
0x3: Divide by 64

Bits 7:5  Reserved, must be kept at reset value.

Bit 4  **SWOENA**: Asynchronous clocking enable for the timestamp counter (read-only)
0: Timestamp counter uses processor clock

Bit 3  **TXENA**: Hardware event packet forwarding enable
Enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2  **SYNCENA**: Synchronization packet transmission enable
If a debugger sets this bit it must also configure the DWT_CTRL register SYNCTAP field in the DWT for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1  **TSENA**: Local timestamp generation enable
0: Disabled
1: Enabled

Bit 0  **ITMENA**: ITM enable
0: Disabled
1: Enabled
### ITM CoreSight peripheral identity register 4 (M7_ITM_PIDR4)

Address offset: 0xFD0  
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:4  **SIZE[3:0]**: Register file size  
0x0: Register file occupies a single 4 Kbyte region  

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

### ITM CoreSight peripheral identity register 0 (M7_ITM_PIDR0)

Address offset: 0xFE0  
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]  
0x01: ITM part number

### ITM CoreSight peripheral identity register 1 (M7_ITM_PIDR1)

Address offset: 0xFE4  
Reset value: 0x0000 00B0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  Reserved, must be kept at reset value.  
Bits 7:0  **JEP106ID[3:0]**  **PARTNUM[11:8]**:  
0x01: ITM part number
Debug infrastructure

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

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

0x1: ITM part number

**ITM CoreSight peripheral identity register 2 (M7_ITM_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4 REVISION[3:0]: Component revision number
0x0: r0p0

Bit 3 JEDEC: JEDEC assigned value
1: Designer ID specified by JEDEC

0x3: Arm® JEDEC code

**ITM CoreSight peripheral identity register 3 (M7_ITM_PIDR3)**

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4 REVAND[3:0]: Metal fix version
0x0: No metal fix

Bits 3:0 CMOD[3:0]: Customer modified
0x0: No customer modifications
**ITM CoreSight component identity register 0 (M7_ITM_CIDR0)**

Address offset: 0xFF0

Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PREAMBLE[7:0]**: Component ID field, bits [7:0]

0x0D: Common ID value

**ITM CoreSight component identity register 1 (M7_ITM_CIDR1)**

Address offset: 0xFF4

Reset value: 0x0000 00E0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class

0x0E: Trace generator component

Bits 3:0  **PREAMBLE[11:8]**: Component ID field, bits [11:8]

0x0: Common ID value

**ITM CoreSight component identity register 2 (M7_ITM_CIDR2)**

Address offset: 0xFF8

Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **CLASS[3:0]**: Component ID field, bits [15:12] - component class

0x0E: Trace generator component

Bits 3:0  **PREAMBLE[19:12]**: Component ID field, bits [11:8]

0x0: Common ID value
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

### ITM CoreSight component identity register 3 (M7_ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

---

**Table 790. Cortex-M7 ITM register map and reset values**

| Offset | Register name | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 79 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|--------|---------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x00   | M7_ITM_STIM0-31 | STIMULUS[31:0] |
| 0x080  | Reserved        |                |
| 0xE00  | M7_ITM_TER      | STIMENA[31:0]  |
| 0xE40  | M7_ITM_TPR      |                |
| 0xE80  | M7_ITM_TCR      | BUSY, TRACEBUSID[6:0], GTSFREQ[1:0], TSPRESCALE[1:0], SWOENA, TXENA, SYNCENA, TSENA, ITMENA |

66.9.11  **Cortex-M7 ITM register map and reset values**

The ITM registers are located at address range 0xE0000000 to 0xE0000FFC, on the AHBD.
### Table 790. Cortex-M7 ITM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
<th>Field 4</th>
<th>Field 5</th>
<th>Field 6</th>
<th>Field 7</th>
<th>Field 8</th>
<th>Field 9</th>
<th>Field 10</th>
<th>Field 11</th>
<th>Field 12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xE84 to 0xFCC</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD0</td>
<td>M7_ITM_PIDR4</td>
<td>SIZE[3:0]</td>
<td>JEP106ID[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>0 0 0 0</td>
<td>0 1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFD4 to 0xFDCC</td>
<td>Reserved</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE0</td>
<td>M7_ITM_PIDR0</td>
<td>PARTNUM[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>0 0 0 0 0 0 0 0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE4</td>
<td>M7_ITM_PIDR1</td>
<td>PARTNUM[1:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>1 0 1 1 0 0 0 0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFE8</td>
<td>M7_ITM_PIDR2</td>
<td>REVISION[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>0 0 0 0 1 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFEC</td>
<td>M7_ITM_PIDR3</td>
<td>REVAND[3:0]</td>
<td>CMOD[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF0</td>
<td>M7_ITM_CIDR0</td>
<td>PREAMBLE[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>0 0 0 0 1 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF4</td>
<td>M7_ITM_CIDR1</td>
<td>CLASS[3:0]</td>
<td>PREAMBLE[11:8]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>1 1 1 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFF8</td>
<td>M7_ITM_CIDR2</td>
<td>PREAMBLE[19:12]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>0 0 0 0 0 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFFC</td>
<td>M7_ITM_CIDR3</td>
<td>PREAMBLE[27:20]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Reset value</td>
<td>- - - - - - - - - - - - - - - - -</td>
<td>1 0 1 1 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
66.9.12 Cortex-M7 breakpoint unit (FPB)

The FPB allows hardware breakpoints to be set. It contains eight comparators which monitor the instruction fetch address and return a breakpoint instruction when a match is detected. The Cortex-M7 FPB does not support flash patch functionality.

66.9.13 Cortex-M7 FPB registers

FPB control register (M7_FPB_CTRL)

Address offset: 0x000
Reset value: 0x1000 0080

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>REV[3:0]: Flash Patch Breakpoint architecture revision.</td>
</tr>
<tr>
<td></td>
<td>0x1: Flash Patch Breakpoint version 2</td>
</tr>
<tr>
<td>27</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>14</td>
<td>NUM_CODE[6:4]: Instruction address comparator number field, three MSBs</td>
</tr>
<tr>
<td></td>
<td>This read-only field holds the three MSBs of the number of instruction address comparators supported.</td>
</tr>
<tr>
<td>11</td>
<td>NUM_LIT[3:0]: Number of literal address comparators supported (read-only).</td>
</tr>
<tr>
<td></td>
<td>0x0: No literal comparators supported.</td>
</tr>
<tr>
<td>7</td>
<td>NUM_CODE[3:0]: Instruction address comparator number field, four LSBs</td>
</tr>
<tr>
<td></td>
<td>This read-only field holds the four LSBs of the number of instruction address comparators supported.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>1</td>
<td>KEY: Write protect key</td>
</tr>
<tr>
<td></td>
<td>A write to M7_FPB_CTRL register is ignored if this bit is not set to 1.</td>
</tr>
<tr>
<td>0</td>
<td>ENABLE: FPB enable</td>
</tr>
<tr>
<td></td>
<td>0: Disable</td>
</tr>
<tr>
<td></td>
<td>1: Enable</td>
</tr>
</tbody>
</table>
FPB comparator registers (M7_FPB_COMPx)

Address offset: 0x008 + x * 0x4 (x = 0 to 7)
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>BPADDR[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
</tbody>
</table>

Bits 31:1: BPADDR[31:1]: Breakpoint address.
This bitfield specifies the breakpoint instruction address.

Bit 0: BE: Breakpoint enable
0: Disabled
1: Enabled

FPB CoreSight peripheral identity register 4 (M7_FPB_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>BPADDR[15:1]</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
<tr>
<td>rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw rw</td>
</tr>
</tbody>
</table>

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

Bits 7:4: SIZE[3:0]: Register file size
0x0: Register file occupies a single 4 Kbyte region

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

FPB CoreSight peripheral identity register 0 (M7_FPB_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 000E

<table>
<thead>
<tr>
<th>PARTNUM[7:0]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r r r r r r</td>
</tr>
<tr>
<td>r r r r r r</td>
</tr>
</tbody>
</table>

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

 Bits 7:4: SIZE[3:0]: Register file size
0x0: Register file occupies a single 4 Kbyte region

 Bits 3:0: JEP106CON[3:0]: JEP106 continuation code
0x4: Arm® JEDEC code
Bits 31:8  Reserved, must be kept at reset value.

Bits 7:0  **PARTNUM[7:0]**: Part number field, bits [7:0]
0x0E: FPB part number

**FPB CoreSight peripheral identity register 1 (M7_FPB_PIDR1)**

Address offset: 0xFE4
Reset value: 0x0000 00B0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r r r r</td>
<td>r r r r</td>
</tr>
</tbody>
</table>

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

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

Bits 3:0  **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: FPB part number

**FPB CoreSight peripheral identity register 2 (M7_FPB_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 000B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r r r r</td>
<td>r r</td>
<td>r r r r</td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3  **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code
FPB CoreSight peripheral identity register 3 (M7_FPB_PIDR3)

Address offset: 0xFEC
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>REVAND[3:0]</td>
<td>CMOD[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:4  REVAND[3:0]: Metal fix version
0x0: No metal fix

Bits 3:0  CMOD[3:0]: Customer modified
0x0: No customer modifications

FPB CoreSight component identity register 0 (M7_FPB_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>PREAMBLE[7:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:0  PREAMBLE[7:0]: Component ID field, bits [7:0]
0x0D: Common ID value

FPB CoreSight component identity register 1 (M7_FPB_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 00E0

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

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

Bits 7:0  PREAMBLE[11:8]: Component ID field, bits [11:8]
**FPB CoreSight component identity register 2 (M7_FPB_CIDR2)**

Address offset: 0xFF8

Reset value: 0x0000 0005

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

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

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

0xE: Trace generator component
0x05: Common ID value

**FPB CoreSight component identity register 3 (M7_FPB_CIDR3)**

Address offset: 0xFFC

Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

0xB1: Common ID value

**FPB CoreSight component identity register 3 (M7_FPB_CIDR3)**

Address offset: 0xFFC

Reset value: 0x0000 00B1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

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

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value
66.9.14 Cortex-M7 FPB register map and reset values

The Cortex-M7 FPB registers are located at address range 0xE0002000 to 0xE0002FFC.

Table 791. Cortex-M7 FPB register map and reset values

| Offset | Register name        | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|--------|----------------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x000  | M7_FPB_CTRL          |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | 0  | 0  | 0  | 1  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0x004  | Reserved             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x008  | M7_FPB_COMPT        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x024  | Reserved             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x028  | M7_FPB_COMP0 to     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x02C  | M7_FPB_COMP7        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD0  | M7_FPB_PIDR4        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFD4  | Reserved             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFD8  | Reserved             |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0xFE0  | M7_FPB_PIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFE4  | M7_FPB_PIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFE8  | M7_FPB_PIDR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFEC  | M7_FPB_PIDR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFF0  | M7_FPB_CIDR0        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFF4  | M7_FPB_CIDR1        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFF8  | M7_FPB_CIDR2        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
| 0xFFC  | M7_FPB_CIDR3        |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        | Reset value          | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  | -  |
66.9.15  **Cortex-M7 embedded trace macrocell (ETM)**

The Cortex-M7 ETM is a CoreSight™ component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M7 core to be traced. In the STM32H7, the ETM is configured for instruction trace only, so data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:
- The number of instructions executed in the same cycle
- Changes in program flow
- The current processor instruction state
- The memory address locations accessed by load and store instructions
- The type, direction and size of a transfer
- Condition code information
- Exception information
- Wait for interrupt state information

For more information, refer to the Arm® CoreSight™ ETM™-M7 technical reference manual [6].
### 66.9.16 Cortex-M7 ETM registers

#### ETM programming control register (M7_ETM_PRGCTL)

Address offset: 0x004  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 0 **EN**: Trace program enable

- 0: Trace unit is disabled
- 1: Trace unit is enabled

#### ETM processor select control register (M7_ETM_PROCSSEL)

Address offset: 0x008  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bit 0 **PROCSEL**: Processor select

This field has no effect since only the Cortex-M7 uses this ETM.

#### ETM status register (M7_ETM_STAT)

Address offset: 0x00C  
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:1 Reserved, must be kept at reset value.
ETM trace configuration register (M7_ETM_CONFIG)

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31:2</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1</td>
<td>PMSTABLE: Programmers model stable</td>
</tr>
<tr>
<td></td>
<td>Indicates whether the ETM registers are stable and can be read.</td>
</tr>
<tr>
<td></td>
<td>0: Registers are not stable</td>
</tr>
<tr>
<td></td>
<td>1: Registers are stable</td>
</tr>
<tr>
<td>Bit 0</td>
<td>IDLE: Trace unit inactive</td>
</tr>
<tr>
<td></td>
<td>0: ETM is not idle</td>
</tr>
<tr>
<td></td>
<td>1: ETM is idle</td>
</tr>
</tbody>
</table>

**ETM trace configuration register (M7_ETM_CONFIG)**

Address offset: 0x010
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bits 31:18</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 17</td>
<td>DV: Data value tracing (read-only)</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>Bit 16</td>
<td>DA: Data address tracing (read-only)</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td>Bits 15:13</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 12</td>
<td>RS: Return stack enable</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bit 11</td>
<td>TS: Global timestamp tracing</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
<tr>
<td>Bits 10:8</td>
<td>COND[2:0]: Conditional instruction tracing</td>
</tr>
<tr>
<td></td>
<td>0x0: Conditional instruction tracing disabled</td>
</tr>
<tr>
<td></td>
<td>0x1: Conditional load instructions are traced</td>
</tr>
<tr>
<td></td>
<td>0x2: Conditional store instructions are traced</td>
</tr>
<tr>
<td></td>
<td>0x3: Conditional load and store instructions are traced</td>
</tr>
<tr>
<td></td>
<td>0x7: All conditional instructions are traced</td>
</tr>
<tr>
<td></td>
<td>Other: Reserved</td>
</tr>
<tr>
<td>Bits 7:5</td>
<td>Reserved, must be kept at reset value.</td>
</tr>
<tr>
<td>Bit 4</td>
<td>CCI: Cycle counting in instruction trace</td>
</tr>
<tr>
<td></td>
<td>0: Disabled</td>
</tr>
<tr>
<td></td>
<td>1: Enabled</td>
</tr>
</tbody>
</table>
ETM event control 0 register (M7_ETM_EVENTCTL0)

Only accepts writes when trace unit is disabled
Address offset: 0x020
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 3</th>
<th>BB: Branch broadcast mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:</td>
<td>Disabled</td>
</tr>
<tr>
<td>1:</td>
<td>Enabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bits 2:1</th>
<th>INSTP0[1:0]: Determines which instructions are P0 instructions (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0:</td>
<td>Only branches are P0 instructions</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Reserved, must be kept at reset value.</th>
</tr>
</thead>
</table>

### Register Layout

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TYPE1</td>
<td>Resource type for event 1</td>
</tr>
<tr>
<td>SEL1[3:0]</td>
<td>Resource / Boolean combined resource pair, for event 1</td>
</tr>
<tr>
<td>TYPE0</td>
<td>Resource type for event 0</td>
</tr>
<tr>
<td>SEL0[3:0]</td>
<td>Resource / Boolean combined resource pair, for event 0</td>
</tr>
</tbody>
</table>

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPE1**: Resource type for event 1
- 0: Single selected resource
- 1: Boolean combined resource pair

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:8 **SEL1[3:0]**: Resource / Boolean combined resource pair, for event 1
- When TYPE1 is 0, selects a single selected resource from 0-15 defined by bits[3:0]
- When TYPE1 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

Bit 7 **TYPE0**: Resource type for event 0
- 0: Single selected resource
- 1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL0[3:0]**: Resource / Boolean combined resource pair for event 0
- When TYPE0 is 0, selects a single selected resource from 0-15 defined by bits[3:0]
- When TYPE0 is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]
### ETM event control 1 register (M7_ETM_EVENTCTL1)

Only accepts writes when trace unit is disabled

Address offset: 0x024

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

Bits 31:13 Reserved, must be kept at reset value.

- **Bit 12 LPOVERRIDE**: Low power state behavior override
  - 0: Low power state normal behavior
  - 1: Entry to low power state does not affect resources and event trace generation

- **Bit 11 ATB**: ATB trigger enable
  - 0: Disabled
  - 1: Enabled

Bits 10:4 Reserved, must be kept at reset value.

- **Bits 3:0 INSTEN[3:0]**: Instruction trace event element enable
  - Each bit corresponds to an event:
    - 0bXXX0: Event 0 does not cause an event element
    - 0bXXX1: Event 0 causes an event element
    - 0bXX0X: Event 1 does not cause an event element
    - 0bXX1X: Event 1 causes an event element
    - 0bX0XX: Event 2 does not cause an event element
    - 0bX1XX: Event 2 causes an event element
    - 0b0XXX: Event 3 does not cause an event element
    - 0b1XXX: Event 3 causes an event element

### ETM stall control register (M7_ETMSTALLCTL)

Only accepts writes when trace unit is disabled

Address offset: 0x02C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
</table>

- **DSTALL**: Debug stall event enable
  - 0: Disabled
  - 1: Enabled

- **ISTALL**: Instruction stall event enable
  - 0: Disabled
  - 1: Enabled

- **Level[1:0]**
Bits 31:10  Reserved, must be kept at reset value.

Bit 9  **DSTALL**: Stall processor based on data trace buffer space
   0: Do not stall processor
   1: Stall processor

Bit 8  **ISTALL**: Stall processor based on instruction trace buffer space
   0: Do not stall processor
   1: Stall processor

Bits 7:4  Reserved, must be kept at reset value.

Bits 3:2  **LEVEL[1:0]**: Stalling threshold level
   A low level minimizes the amount of processor stalling, with a higher risk of FIFO overflow. A high level minimizes the risk of FIFO overflow but increases the amount of processor stalling.

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

**ETM global timestamp control register (M7_ETM_TSCTL)**

Only accepts writes when trace unit is disabled

Address offset: 0x030

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>TYPE</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bit 7  **TYPE**: Resource type for time stamp insertion
   0: Single selected resource
   1: Boolean combined resource pair

Bits 6:4  Reserved, must be kept at reset value.

Bits 3:0  **SEL[3:0]**: Resource / Boolean combined resource pair
   When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]
   When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

**ETM synchronization period register (M7_ETM_SYNCP)**

Address offset: 0x034

Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>
Bits 31:5  Reserved, must be kept at reset value.

Bits 4:0  **PERIOD[4:0]**: Trace bytes between synchronization requests
  Defines the number of bytes of trace information between trace synchronization requests.
  0xA: 1024 bytes

**ETM cycle count control register (M7_ETM_CCCTL)**
Address offset: 0x038
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit  9</th>
<th>Bit  8</th>
<th>Bit  7</th>
<th>Bit  6</th>
<th>Bit  5</th>
<th>Bit  4</th>
<th>Bit  3</th>
<th>Bit  2</th>
<th>Bit  1</th>
<th>Bit  0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ETM trace ID register (M7_ETM_TRACEID)**
Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit  9</th>
<th>Bit  8</th>
<th>Bit  7</th>
<th>Bit  6</th>
<th>Bit  5</th>
<th>Bit  4</th>
<th>Bit  3</th>
<th>Bit  2</th>
<th>Bit  1</th>
<th>Bit  0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:12  Reserved, must be kept at reset value.

Bits 11:0  **THRESHOLD[11:0]**: Threshold value for instruction trace cycle counting
  The threshold represents the minimum interval between cycle count trace packets.
  0x0: Reserved
  Other: Threshold

**ETM trace ID register (M7_ETM_TRACEID)**
Address offset: 0x040
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>Bit 30</th>
<th>Bit 29</th>
<th>Bit 28</th>
<th>Bit 27</th>
<th>Bit 26</th>
<th>Bit 25</th>
<th>Bit 24</th>
<th>Bit 23</th>
<th>Bit 22</th>
<th>Bit 21</th>
<th>Bit 20</th>
<th>Bit 19</th>
<th>Bit 18</th>
<th>Bit 17</th>
<th>Bit 16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit  9</th>
<th>Bit  8</th>
<th>Bit  7</th>
<th>Bit  6</th>
<th>Bit  5</th>
<th>Bit  4</th>
<th>Bit  3</th>
<th>Bit  2</th>
<th>Bit  1</th>
<th>Bit  0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:7  Reserved, must be kept at reset value.

Bits 6:0  **TRACEID[6:0]**: Trace ID
  0x00: Reserved
  0x01 to 0x6F: Valid ID
  0x70 to 0x7F: Reserved
ETM ViewInst main control register (M7_ETM_VICTL)

Address offset: 0x080
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

- **Bit 19 EXLEVEL_S3**: Trace disable, exception level 3
  - Disables tracing in the specified exception level in Secure state for exception level 3.
  - 0: Enable ViewInst in this exception level
  - 1: Disable ViewInst in this exception level

Bits 18:17 Reserved, must be kept at reset value.

- **Bit 16 EXLEVEL_S0**: Trace disable, exception level 0
  - Disables tracing in the specified exception level in Secure state for exception level 0.
  - 0: Enable ViewInst in this exception level
  - 1: Disable ViewInst in this exception level

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

- **Bit 11 TRCERR**: Tracing of system error exception
  - Selects whether a system error exception must always be traced.
  - 0: System error exception is traced only if the instruction or exception immediately before the system error exception is traced
  - 1: System error exception is always traced regardless of the value of ViewInst

- **Bit 10 TRCRESET**: Tracing of reset exception
  - Selects whether a reset exception must always be traced.
  - 0: Reset exception is traced only if the instruction or exception immediately before the reset exception is traced
  - 1: Reset exception is always traced regardless of the value of ViewInst

- **Bit 9 SSSTATUS**: Current status of the start/stop logic
  - 0: Stop state
  - 1: Started state

- **Bit 8** Reserved, must be kept at reset value.
Bit 7 **TYPE**: Resource type  
0: Single selected resource  
1: Boolean combined resource pair

Bits 6:4 Reserved, must be kept at reset value.

Bits 3:0 **SEL[3:0]**: Resource / Boolean combined resource pair  
When TYPE is 0, selects a single selected resource from 0-15 defined by bits[3:0]  
When TYPE is 1, selects a Boolean combined resource pair from 0-7 defined by bits[2:0]

### ETM ViewInst start/stop control register (M7_ETM_VISSCTL)

Address offset: 0x088  
Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **STOP[7:0]**: Single address comparator selector to stop trace  
Defines the single address comparators to stop trace with the ViewInst Start/Stop control.
One bit is provided for each implemented single address comparator.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **START[7:0]**: Single address comparator selector to start trace  
Defines the single address comparators to start trace with the ViewInst Start/Stop control.
One bit is provided for each implemented single address comparator.

### ETM ViewInst start/stop processor comparator control register  
(M7_ETM_VIPCSSCTL)

Address offset: 0x08C  
Reset value: 0x0000 0000

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

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

Bits 16:0 **START[3:0]**: Single address comparator selector to start trace  
Defines the single address comparators to start trace with the ViewInst Start/Stop control.
One bit is provided for each implemented single address comparator.
Bits 31:20  Reserved, must be kept at reset value.

Bits 19:16  **STOP[3:0]**: Processor comparator input selector to stop trace
Selects which processor comparator inputs are in use with ViewInst start-stop control, for the purpose of stopping trace. One bit is provided for each processor comparator input.

Bits 15:4  Reserved, must be kept at reset value.

Bits 3:0  **START[3:0]**: Processor comparator input selector to start trace
Selects which processor comparator inputs are in use with ViewInst start-stop control, for the purpose of starting trace. One bit is provided for each processor comparator input.

**ETM counter reload value register (M7_ETM_CNTRLDV)**

Address offset: 0x140
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**VALUE[15:0]**

Bits 31:16  Reserved, must be kept at reset value.

Bits 15:0  **VALUE[15:0]**: Counter reload value
This value is loaded into the counter each time the reload event occurs.

**ETM ID register 8 (M7_ETM_IDR8)**

Address offset: 0x180
Reset value: 0x0000 0002

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**MAXSPEC[31:16]**

Bits 31:0  **MAXSPEC[31:0]**: Maximum speculation depth
Indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.

0x2: Maximum trace speculation depth is 2
## ETM ID register 9 (M7_ETM_IDR9)
Address offset: 0x184
Reset value: 0x0000 0000

### NUMP0KEY[31:16]

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### NUMP0KEY[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used
0x0: No P0 keys used in instruction trace only configuration

## ETM ID register 10 (M7_ETM_IDR10)
Address offset: 0x188
Reset value: 0x0000 0000

### NUMP1KEY[31:16]

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### NUMP1KEY[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **NUMP1KEY[31:0]**: Total number of P1 right-hand keys
Indicates the total number of P1 right-hand keys, including normal and special keys.
0x0: No P1 keys used in instruction trace only configuration

## ETM ID register 11 (M7_ETM_IDR11)
Address offset: 0x18C
Reset value: 0x0000 0000

### NUMP1SPC[31:16]

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

### NUMP1SPC[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
</tbody>
</table>

Bits 31:0 **NUMP1SPC[31:0]**: Total number of special P1 right-hand keys used
0x0: No special P1 keys used
ETM ID register 12 (M7_ETM_IDR12)
Address offset: 0x190
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

NUMCONDKEY[31:16]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 NUMCONDKEY[31:0]: Indicates the total number of conditional instruction right-hand keys, including normal and special keys.
0x1: One conditional instruction right hand-key implemented

ETM ID register 13 (M7_ETM_IDR13)
Address offset: 0x194
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

NUMCONDSPC[31:16]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NUMCONDSPC[15:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0 NUMCONDSPC[31:0]: Number of special conditional instruction right-hand keys
0x0: No special conditional instruction right-hand keys implemented

ETM implementation specific register 0 (M7_ETM_IMSPEC0)
Address offset: 0x1C0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SUPPORT[3:0]

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 3:0 SUPPORT[3:0]: Support for implementation specific extensions
0x0: No implementation specific extensions are supported
**ETM ID register 0 (M7_ETM_IDR0)**

Address offset: 0x1E0

Reset value: 0x0C00 1EE1

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:30  Reserved, must be kept at reset value.

Bit 29  **COMM OPT**: Meaning of the commit field in some packets
0: Commit mode 0

Bits 28:24  **TSSIZE[4:0]**: Global timestamp size
0x08: Maximum of 64-bit global timestamp implemented

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

Bits 16:15  **QSUPP[1:0]**: Q element support
0x0: Q elements not supported

Bit 14  Reserved, must be kept at reset value.

Bits 13:12  **CONDTYPE[1:0]**: Conditional result tracing type
0x1: APSR condition flag values traced

Bits 11:10  **NUMEVENT[1:0]**: Number of events supported in the trace
0x1: Two events supported for instruction only configuration

Bit 9  **RETSTACK**: Return stack support
1: Two entry return stack supported

Bit 8  Reserved, must be kept at reset value.

Bit 7  **TRCCCCI**: Support for cycle counting in the instruction trace
1: Cycle counting in the instruction trace is implemented

Bit 6  **TRCCOND**: Support for conditional instruction tracing
1: Conditional instruction trace is implemented

Bit 5  **TRCBB**: Support for branch broadcast tracing
1: Branch broadcast trace is implemented

Bits 4:0  Reserved, must be kept at reset value.
ETM ID register 1 (M7_ETM_IDR1)
Address offset: 0x1E4
Reset value: 0x4100 F401

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
<td>RES</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:24 DESIGNER[7:0]: Trace unit designer entity
0x41: Arm®

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 TRCARCHMAJ[3:0]: Major trace unit architecture version number
0x4: ETM v4

Bits 7:4 TRCARCHMIN[3:0]: Minor trace unit architecture version number
0x0: Minor version 0

Bits 3:0 REVISION[3:0]: Implementation revision number
0x1: Rev 1

ETM ID register 2 (M7_ETM_IDR2)
Address offset: 0x1E8
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 CCSIZE[3:0]: Cycle counter size
Indicates the size of the cycle counter in bits minus 12.
0x0: Cycle counter is 12 bits

Bits 24:20 DVSIZE[4:0]: Data value size in bytes
0x0: Data value size is not supported in instruction only configuration

Bits 19:15 DASIZE[4:0]: Data address size in bytes
0x0: Data address size is not supported in instruction only configuration
ETM ID register 3 (M7_ETM_IDR3)

Address offset: 0x1EC
Reset value: 0x0509 0004

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31 30 29 28</td>
<td>27 26 25 24</td>
<td>23 22 21</td>
<td>20 19 18</td>
<td>17 16</td>
<td>15 14</td>
<td>13 12</td>
<td>11 10</td>
<td>9</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31 NOOVERFLOW: Support of NOOVERFLOW
Indicates whether the NOOVERFLOW of trace stall control is implemented.
0: Not implemented

Bits 30:28 NUMPROC[2:0]: Number of processors available for tracing
0x0: Only one processor can be traced

Bit 27 SYSSTALL: System support for stall control of the processor
0: Not supported

Bit 26 STALLCTL: Stall control support
1: Trace stall control (TRCSTALLCTL) is implemented

Bit 25 SYNCP: Trace synchronization period support
0: TRCSYNCP is read-only for instruction trace only configuration; the trace synchronization period is fixed

Bit 24 TRCERR: Support of TRCVICTLR.TRCERR
Indicates whether TRCVICTLR.TRCERR is implemented.
0x4: 32-bit maximum address size

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

Bits 19:16 EXLEVEL_S[3:0]: Support of privilege levels
Privilege levels are implemented; one bit for each level.
0x9: Privilege levels Thread and Handler are implemented

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

Bits 11:0 CCITMIN[11:0]: Instruction trace cycle counting minimum threshold
0x4: Minimum threshold is 4 instruction trace cycle
ETM ID register 4 (M7_ETM_IDR4)

Address offset: 0x1F0
Reset value: 0x0001 4000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Bits 31:28 NUMVMIDC[3:0]: Number of Virtual Machine ID comparators implemented
0x0: None

*Bits 27:24 NUMCIDC[3:0]: Number of Context ID comparators implemented
0x0: None

*Bits 23:20 NUMSSCC[3:0]: Number of single-shot comparator controls implemented
0x0: None

*Bits 19:16 NUMRSPAIR[3:0]: Number of resource selection pairs implemented
0x1: None

*Bits 15:12 NUMPC[3:0]: Number of processor comparator inputs implemented
0x4: Four

*Bits 11:9 Reserved, must be kept at reset value.

*Bit 8 SUPPADC: Support of data address comparisons
0: Not implemented

*Bits 7:4 NUMDVC[3:0]: Number of data value comparators implemented
0x0: None

*Bits 3:0 NUMACPAIRS[3:0]: Number of address comparator pairs implemented
0x0: None

ETM ID register 5 (M7_ETM_IDR5)

Address offset: 0x1F4
Reset value: 0x90C7 0402

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

*Bits 31:28 REDFUCNTR: Reserved

*Bits 28:27 NUMCNTR[2:0]: Number of instruction counter comparators

*Bits 26:22 NUMSEQSTATE[2:0]: Number of sequence state comparators

*Bit 21 Res.: Reserved, must be kept at reset value.

*Bit 20 LPOVE: Support of load operation comparison
0: Not implemented

*Bit 19 RRIDE: Support of register read instruction comparison
0: Not implemented

*Bit 18 ATBTRIG: Support of address trace buffer trigger
0: Not implemented

*Bits 17:0 TRACEIDSIZE[5:0]: Trace ID size
0x0: None
**ETM resource selection register 2 (M7_ETM_RSCTL2)**

Address offset: 0x208

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>Bit 31</th>
<th>REDFUNCNTR: Support of reduced function counter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1: Implemented</td>
</tr>
<tr>
<td>Bits 30:28</td>
<td>NUMCNT[2:0]: Number of counters implemented</td>
</tr>
<tr>
<td></td>
<td>0x1: One counter implemented</td>
</tr>
<tr>
<td>Bits 27:25</td>
<td>NUMSEQSTATE[2:0]: Number of sequencer states implemented</td>
</tr>
<tr>
<td></td>
<td>0x0: None</td>
</tr>
</tbody>
</table>

Bit 24  Reserved, must be kept at reset value.

Bit 23  LPOVERRIDE: Support of low-power state override

1: Implemented

Bit 22  ATBTRIG: Support of ATB trigger

1: Implemented

Bits 21:16  TRACEDSVSIZE[5:0]: Number of trace ID bits

0x07: Seven-bit trace ID implemented.

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

Bits 11:9  NUMEXTINSEL[2:0]: Number of implemented external input selectors

0x2: Two external input selectors implemented

Bits 8:0  NUMEXTIN[8:0]: Number of implemented external inputs

0x2: Two external inputs implemented

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

Bit 21  PAIRINV: Inversion of result of a combined pair of resources

0: Not inverted
1: Inverted

Bit 20  INV: Inversion of the selected resources

0: Not inverted
1: Inverted

Bit 19  Reserved, must be kept at reset value.
Bits 18:16  **GROUP[2:0]**: Selects a group of resources

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **SELECT[7:0]**: Resource selector from desired group

Selects one or more resources from the desired group. One bit is provided per resource from the group.

**ETM resource selection register 3 (M7_ETM_RSCTL3)**

Address offset: 0x20C

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:21  Reserved, must be kept at reset value.

Bit 20  **INV**: Inversion of the selected resources

0: Not inverted
1: Inverted

Bit 19  Reserved, must be kept at reset value.

Bits 18:16  **GROUP[2:0]**: Selects a group of resources

Bits 15:8  Reserved, must be kept at reset value.

Bits 7:0  **SELECT[7:0]**: Resource selector from desired group

Selects one or more resources from the desired group. One bit is provided per resource from the group.

**ETM single-shot comparator control register 0 (M7_ETM_SSCC0)**

Address offset: 0x280

Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>rw</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Debug infrastructure RM0477

ETM single-shot comparator status register 0 (M7_ETM_SSCS0)
Address offset: 0x2A0
Reset value: 0x0000 0001

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>rw</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bit 31  **STATUS**: Single-shot status
This indicates whether any of the selected comparators have matched. If SSCC0.RST is set to 0, the STATUS bit must be written with 0 in order to enable single-shot comparator control.

0: No match occurred
1: Match has occurred at least once.

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

Bit 2  **DV**: Data value comparator support
0: Single-shot data value comparisons not supported

Bit 1  **DA**: Data address comparator support
0: Single-shot data address comparisons not supported

Bit 0  **INST**: Instruction address comparator support
1: Single-shot instruction address comparisons supported

ETM single-shot processor comparator input control register (M7_ETM_SSPCIC0)
Address offset: 0x2C0
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC[7:0]</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
<td>rw</td>
</tr>
</tbody>
</table>

3756/3791 RM0477 Rev 6
RM0477 Debug infrastructure

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

Bits 7:0 **PC[7:0]:** Comparator input selector for single-shot control
Selects one or more processor comparator inputs for single-shot control. One bit is provided for each processor comparator input.

**ETM power-down control register (M7_ETM_PDC)**

Address offset: 0x310
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 3 **PU: Power up request
Request to maintain power to the ETM and access to the trace registers.

0: Power not requested
1: Power requested

Bits 2:0 Reserved, must be kept at reset value.

**ETM power-down status register (M7_ETM_PDS)**

Address offset: 0x314
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Bit 1 **STICKYPD: Sticky power-down state
This bit is set to 1 when power to the ETM registers is removed, to indicate that programming state has been lost. It is cleared after a read of the TRCPDSR.

0: Trace register power uninterrupted since the last read of PDS register
1: Trace register power interrupted since the last read of PDS register

**Bit 0 **POWER: ETM powered up
1: ETM is powered up; all registers are accessible
ETM claim tag set register (M7_ETM_CLAIMSET)

Address offset: 0xFA0
Reset value: 0x0000 000F

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  CLAIMSET[3:0]: Set claim tag bits

Write:
- 0000: No effect
- xxx1: Set bit 0
- xx1x: Set bit 1
- x1xx: Set bit 2
- 1xxx: Set bit 3

Read:
- 0xF: Indicates there are four bits in claim tag

ETM claim tag clear register (M7_ETM_CLAIMCLR)

Address offset: 0xFA4
Reset value: 0x0000 0000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:4  Reserved, must be kept at reset value.
Bits 3:0  CLAIMCLR[3:0]: Reset claim tag bits

Write:
- 0000: No effect
- xxx1: Clear bit 0
- xx1x: Clear bit 1
- x1xx: Clear bit 2
- 1xxx: Clear bit 3

Read: Returns current value of claim tag
**ETM lock access register (M7_ETM_LAR)**

Address offset: 0xFB0  
Reset value: 0xXXXX XXXX

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**ACCESS_W[31:16]**

Bits 31:0 **ACCESS_W[31:0]**: ETM register write access  
Enables write access to some ETM registers by the processor (debuggers do not need to unlock the component)

0xC5ACCE55: Enable write access  
Other values: Disable write access

**ETM lock status register (M7_ETM_LSR)**

Address offset: 0xFB4  
Reset value: 0x0000 0003

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ACCESS_W[15:0]**

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
<td>w</td>
</tr>
</tbody>
</table>

**ACCESS_W[15:0]**

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

- **Bit 2** **LOCKTYPE**: Size of the M7_ETM_LAR register
  
  0: 32-bit

- **Bit 1** **LOCKGRANT**: Current status of lock
  
  This bit always returns zero when read by an external debugger.
  
  0: Write access is permitted  
  1: Write access is blocked. Only read access is permitted.

- **Bit 0** **LOCKEXIST**: Existence of lock control mechanism
  
  The bit indicates whether a lock control mechanism exists. It always returns zero when read by an external debugger.
  
  0: No lock control mechanism exists  
  1: Lock control mechanism is implemented
ETM authentication status register (M7_ETM_AUTHSTAT)

Address offset: 0xFB8
Reset value: 0x0000 000A

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Bits 7:6  **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4  **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2  **NSNID[1:0]**: Security level for non-secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0  **NSID[1:0]**: Security level for non-secure invasive debug
0x2: Disabled
0x3: Enabled

ETM CoreSight device architecture register (M7_ETM_DEVARrch)

Address offset: 0xFB C
Reset value: 0x4770 4A13

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARCHITECT[10:0]</td>
<td>PRESENT</td>
<td>REVISION[3:0]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:21  **ARCHITECT[10:0]**: Component architect
0x23B: Arm®

Bit 20  **PRESENT**: Indicates the presence of this register
1: Present

Bits 19:16  **REVISION[3:0]**: Architecture revision
0x0: Rev 0

Bits 15:0  **ARCHID[15:0]**: Architecture ID
0x4A13: ETMv4 component
ETM CoreSight device type identity register (M7_ETM_DEVTYPE)

Address offset: 0xFCC
Reset value: 0x0000 0013

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4 **SUBTYPE[3:0]**: Device sub-type identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (M7_ETM_PIDR4)

Address offset: 0xFD0
Reset value: 0x0000 0004

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4 Kbyte region

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

ETM CoreSight peripheral identity register 0 (M7_ETM_PIDR0)

Address offset: 0xFE0
Reset value: 0x0000 0075

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 **PARTNUM[7:0]**
**ETM CoreSight peripheral identity register 1 (M7_ETM_PIDR1)**

Address offset: 0xFE4
Reset value: 0x0000 00B9

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0  **PARTNUM[7:0]**: Part number field, field [7:0]
0x75: ETM part number

**JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm® JEDEC code

**PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETM part number

**ETM CoreSight peripheral identity register 2 (M7_ETM_PIDR2)**

Address offset: 0xFE8
Reset value: 0x0000 001B

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4  **REVISION[3:0]**: Component revision number
0x1: rev 1

Bit 3  **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0  **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm® JEDEC code
ETM CoreSight peripheral identity register 3 (M7_ETM_PIDR3)

Address offset: 0xFEC
Reset value: 0x00000000

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:4: REVAND[3:0]: Metal fix version
0x0: No metal fix

Bits 3:0: CMOD[3:0]: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (M7_ETM_CIDR0)

Address offset: 0xFF0
Reset value: 0x0000 000D

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0: PREAMBLE[7:0]: Component ID field, bits [7:0]
0xD: Common ID value

ETM CoreSight component identity register 1 (M7_ETM_CIDR1)

Address offset: 0xFF4
Reset value: 0x0000 0090

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Bits 7:0: PREAMBLE[11:8]: Component ID field, bits [7:0]
ETM CoreSight component identity register 2 (M7_ETM_CIDR2)
Address offset: 0xFF8
Reset value: 0x0000 0005

ETM CoreSight component identity register 3 (M7_ETM_CIDR3)
Address offset: 0xFFC
Reset value: 0x0000 00B1
# 66.9.17 Cortex-M7 ETM register map and reset values

The ETM registers are accessed by the debugger via the Cortex-M7 PPB, at address range 0xE0041000 to 0xE0041FFC.

**Table 792. Cortex-M7 ETM register map and reset values**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Bits</th>
<th>Description</th>
<th>Reset value</th>
<th>EN</th>
<th>PROCSEL[2:0]</th>
<th>PMSEL[2:0]</th>
<th>IDE</th>
<th>IDLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x004</td>
<td>M7_ETM_PRGCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x008</td>
<td>M7_ETM_PROCSEL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x00C</td>
<td>M7_ETM_STAT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x010</td>
<td>M7_ETM_CONFIG</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x014</td>
<td>M7_ETM_EVENTCTL0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01C</td>
<td>M7_ETM_EVENTCTL1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x020</td>
<td>M7_ETMSTALLCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x030</td>
<td>M7_ETM_TSCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x034</td>
<td>M7_ETM_SYNC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x038</td>
<td>M7_ETM_CCCTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The table provides a detailed view of the register map and reset values for the Cortex-M7 ETM, including bit positions and reset values for each register.
<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x040</td>
<td>M7_ETM_TRACEID</td>
<td>TRACEID[6:0]</td>
</tr>
<tr>
<td>0x044   to 0x07C</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x080</td>
<td>M7_ETM_VICTL</td>
<td></td>
</tr>
<tr>
<td>0x088</td>
<td>M7_ETM_VISSCTL</td>
<td>STOP[7:0]</td>
</tr>
<tr>
<td>0x08C</td>
<td>M7_ETM_VISSCTRL</td>
<td>START[3:0]</td>
</tr>
<tr>
<td>0x090   to 0x093</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x140</td>
<td>M7_ETM_CNTRLDV</td>
<td>VALUE[15:0]</td>
</tr>
<tr>
<td>0x180</td>
<td>M7_ETM_IDR8</td>
<td>MAXSPEC[31:0]</td>
</tr>
<tr>
<td>0x184</td>
<td>M7_ETM_IDR9</td>
<td>NUMP0KEY[31:0]</td>
</tr>
<tr>
<td>0x188</td>
<td>M7_ETM_IDR10</td>
<td>NUMP1KEY[31:0]</td>
</tr>
<tr>
<td>0x18C</td>
<td>M7_ETM_IDR11</td>
<td>NUMP1SPC[31:0]</td>
</tr>
<tr>
<td>0x190</td>
<td>M7_ETM_IDR12</td>
<td>NUMCONDKEY[31:0]</td>
</tr>
<tr>
<td>0x194</td>
<td>M7_ETM_IDR13</td>
<td>NUMCONDSPC[31:0]</td>
</tr>
<tr>
<td>0x198   to 0x1BC</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x1C0</td>
<td>M7_ETM_IMSPEC0</td>
<td>SUPPORT[3:0]</td>
</tr>
<tr>
<td>0x1E0</td>
<td>M7_ETM_IDR0</td>
<td>TSIZE[4:0]</td>
</tr>
</tbody>
</table>

**Table 792. Cortex-M7 ETM register map and reset values (continued)**

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Reset values:**
- 0x040: M7_ETM_TRACEID
- 0x080: M7_ETM_VICTL
- 0x180: M7_ETM_IDR8
- 0x1C0: M7_ETM_IMSPEC0

**Values:**
- 0x040: TRACEID[6:0] = 00000000
- 0x080: M7_ETM_VICTL
- 0x088: M7_ETM_VISSCTL
- 0x08C: M7_ETM_VISSCTRL
- 0x140: M7_ETM_CNTRLDV
- 0x180: M7_ETM_IDR8
- 0x184: M7_ETM_IDR9
- 0x188: M7_ETM_IDR10
- 0x18C: M7_ETM_IDR11
- 0x190: M7_ETM_IDR12
- 0x194: M7_ETM_IDR13
- 0x1C0: M7_ETM_IMSPEC0

**Reserved:**
- 0x044 to 0x07C
- 0x090 to 0x093
- 0x098 to 0x09B
- 0x198 to 0x1BC

**Register descriptions:**
- **M7_ETM_TRACEID**: TRACEID[6:0]
- **M7_ETM_VICTL**: Various registers with different bitfields
- **M7_ETM_VISSCTL**: STOP[7:0]
- **M7_ETM_VISSCTRL**: START[3:0]
- **M7_ETM_CNTRLDV**: VALUE[15:0]
- **M7_ETM_IDR8**: MAXSPEC[31:0]
- **M7_ETM_IDR9**: NUMP0KEY[31:0]
- **M7_ETM_IDR10**: NUMP1KEY[31:0]
- **M7_ETM_IDR11**: NUMP1SPC[31:0]
- **M7_ETM_IDR12**: NUMCONDKEY[31:0]
- **M7_ETM_IDR13**: NUMCONDSPC[31:0]
- **M7_ETM_IMSPEC0**: SUPPORT[3:0]
- **M7_ETM_IDR0**: TSIZE[4:0]

**Values:**
- 0x040: TRACEID[6:0] = 00000000
- 0x080: M7_ETM_VICTL
- 0x088: M7_ETM_VISSCTL
- 0x08C: M7_ETM_VISSCTRL
- 0x140: M7_ETM_CNTRLDV
- 0x180: M7_ETM_IDR8
- 0x184: M7_ETM_IDR9
- 0x188: M7_ETM_IDR10
- 0x18C: M7_ETM_IDR11
- 0x190: M7_ETM_IDR12
- 0x194: M7_ETM_IDR13
- 0x1C0: M7_ETM_IMSPEC0

**Reserved:**
- 0x044 to 0x07C
- 0x090 to 0x093
- 0x098 to 0x09B
- 0x198 to 0x1BC

**Register descriptions:**
- **M7_ETM_TRACEID**: TRACEID[6:0]
- **M7_ETM_VICTL**: Various registers with different bitfields
- **M7_ETM_VISSCTL**: STOP[7:0]
- **M7_ETM_VISSCTRL**: START[3:0]
- **M7_ETM_CNTRLDV**: VALUE[15:0]
- **M7_ETM_IDR8**: MAXSPEC[31:0]
- **M7_ETM_IDR9**: NUMP0KEY[31:0]
- **M7_ETM_IDR10**: NUMP1KEY[31:0]
- **M7_ETM_IDR11**: NUMP1SPC[31:0]
- **M7_ETM_IDR12**: NUMCONDKEY[31:0]
- **M7_ETM_IDR13**: NUMCONDSPC[31:0]
- **M7_ETM_IMSPEC0**: SUPPORT[3:0]
- **M7_ETM_IDR0**: TSIZE[4:0]
| Offset   | Register name  | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
|---------|----------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 0x1E4  | M7_ETM_IDR1    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | - | - | - | - | - | - | - | - | - | - | - | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0x1E8  | M7_ETM_IDR2    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0x1EC  | M7_ETM_IDR3    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | - | - | - | - | 1 | 0 | 0 | 1 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0x1F0  | M7_ETM_IDR4    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x1F4  | M7_ETM_IDR5    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | - | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | - | - | - | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0x1F8 to 0x204 | Reserved     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x208  | M7_ETM_RSCTL2  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | - | - | - | - | - | - | - | - | - | - | 0 | 0 | 0 | 0 | 0 | - | - | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x20C  | M7_ETM_RSCTL3  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | - | - | - | - | - | - | - | - | - | - | 0 | - | 0 | 0 | 0 | - | - | - | - | - | - | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0x210 to 0x27C | Reserved     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
| 0x280  | M7_ETM_SS0CC0  |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|        |                | Reset value | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| 0x284 to 0x29C | Reserved     |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Table 792. Cortex-M7 ETM register map and reset values (continued)
## Table 792. Cortex-M7 ETM register map and reset values (continued)

<table>
<thead>
<tr>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
<th>Offset</th>
<th>Register name</th>
<th>Reset value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x2A0</td>
<td>M7_ETM_SSCS0</td>
<td>0x00000000</td>
<td>0x2C0</td>
<td>M7_ETM_SSPCIC0</td>
<td>0x00000000</td>
<td>0x2C4</td>
<td>M7_ETM_PDC</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x2A4 to 0x2BC</td>
<td>Reserved</td>
<td></td>
<td>0x2C0</td>
<td>M7_ETM_SSPCIC0</td>
<td>0x00000000</td>
<td>0x2C4</td>
<td>M7_ETM_PDC</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x310</td>
<td>M7_ETM_PDC</td>
<td>0x00000000</td>
<td>0x314</td>
<td>M7_ETM_PDS</td>
<td>0x00000000</td>
<td>0x318 to 0x39C</td>
<td>Reserved</td>
<td>0x310</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x31B to 0x39C</td>
<td>Reserved</td>
<td></td>
<td>0x314</td>
<td>M7_ETM_PDS</td>
<td>0x00000000</td>
<td>0x318 to 0x39C</td>
<td>Reserved</td>
<td>0x310</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFA0</td>
<td>M7_ETM _CLAIMSET</td>
<td>0x00000000</td>
<td>0xFA4</td>
<td>M7_ETM _CLAIMCLR</td>
<td>0x00000000</td>
<td>0xFB0</td>
<td>M7_ETM _LAR ACCESS_W</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB0</td>
<td>M7_ETM_LAR</td>
<td>0x00000000</td>
<td>0xFB4</td>
<td>M7_ETM _LSR</td>
<td>0x00000000</td>
<td>0xFB8</td>
<td>M7_ETM _AUTHSTAT</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFB8</td>
<td>M7_ETM _AUTHSTAT</td>
<td>0x00000000</td>
<td>0xFB8</td>
<td>M7_ETM _AUTHSTAT</td>
<td>0x00000000</td>
<td>0xFB8</td>
<td>M7_ETM _AUTHSTAT</td>
<td>0x00000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xFC0 to 0xFC8</td>
<td>Reserved</td>
<td></td>
<td>0xFC0 to 0xFC8</td>
<td>Reserved</td>
<td></td>
<td>0xFC0 to 0xFC8</td>
<td>Reserved</td>
<td></td>
</tr>
</tbody>
</table>
### Table 792. Cortex-M7 ETM register map and reset values (continued)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0xFCC</td>
<td>M7_ETM_DEVTYPE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00000000</td>
<td>0x0100</td>
<td>0xFDC</td>
<td></td>
<td></td>
<td></td>
<td>0xFD0</td>
<td>M7_ETM_PIDR4</td>
<td>JEP106ID[3:0]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFD4</td>
<td>M7_ETM_PIDR1</td>
<td>PARTNUM[1:8]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFE0</td>
<td>M7_ETM_PIDR2</td>
<td>JEP106ID[6:4]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFE4</td>
<td>M7_ETM_PIDR3</td>
<td>CMOD[3:0]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFF0</td>
<td>M7_ETM_CIDR0</td>
<td>PREAMBLE[7:0]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFF4</td>
<td>M7_ETM_CIDR1</td>
<td>PREAMBLE[11:8]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFF8</td>
<td>M7_ETM_CIDR2</td>
<td>PREAMBLE[19:12]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0xFFC</td>
<td>M7_ETM_CIDR3</td>
<td>PREAMBLE[27:20]</td>
<td>0x0000000000000000</td>
<td></td>
</tr>
</tbody>
</table>

#### 66.9.18 Cortex-M7 cross trigger interface (CTI)

See [Section 66.6.4](#).
66.10 References for debug infrastructure

1. IHI 0031C (ID080813) - Arm® Debug Interface Architecture Specification ADlv5.0 to ADlv5.2, Issue C
2. DDI 0480F (ID100313) - Arm® CoreSight™ SoC-400 r3p2 Technical Reference Manual, Issue G
3. DDI 0461B (ID010111) - Arm® CoreSight™ Trace Memory Controller r0p1 Technical Reference Manual, Issue B
5. DDI 0403D (ID100710) - Arm®v7-M Architecture Reference Manual, Issue E.b
6. DDI 0494-2a (ID062813) - Arm® CoreSight™ ETM™-M7 r0p1 Technical Reference Manual, Issue D
67 Device electronic signature

The electronic signature is stored in the Flash memory area. It can be read using the JTAG/SWD or the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match its interface to the characteristics of the STM32H7Rx/7Sx microcontrollers.

67.1 Unique device ID register (96 bits)

The unique device identifier is ideally suited:
- for use as serial numbers (for example USB string serial numbers or other end applications)
- for use as security keys in order to increase the security of code in Flash memory while using and combining this unique ID with software cryptographic primitives and protocols before programming the internal Flash memory
- to activate secure boot processes, etc.

The 96-bit unique device identifier provides a reference number which is unique for any device and in any context. These bits can never be altered by the user.

The 96-bit unique device identifier can also be read in single bytes/half-words/words in different ways and then be concatenated using a custom algorithm.

Base address: 0x08FF F800
Address offset: 0x00
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>U_ID[31:16]</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0</td>
</tr>
</tbody>
</table>

Bits 31:0 U_ID[31:0]: X and Y coordinates on the wafer
Address offset: 0x04
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_ID[63:48]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:8  **U_ID[63:40]**: LOT_NUM[23:0]
Lot number (ASCII encoded)

Bits 39:32  **U_ID[63:32]**: 63:32 unique ID bits
Wafer number (8-bit unsigned number)

Address offset: 0x08
Read only = 0xXXXX XXXX where X is factory-programmed

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_ID[95:80]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>U_ID[79:64]</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td>r</td>
<td></td>
</tr>
</tbody>
</table>

Bits 31:0  **U_ID[95:64]**: LOT_NUM[55:24]
Lot number (ASCII encoded)
67.2 Package data register

Base address: 0x08FF F80C
Address offset: 0x00
Read only = 0xXXXXX where X is factory-programmed

<table>
<thead>
<tr>
<th>31</th>
<th>30</th>
<th>29</th>
<th>28</th>
<th>27</th>
<th>26</th>
<th>25</th>
<th>24</th>
<th>23</th>
<th>22</th>
<th>21</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PKG_CODE</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **PKG_CODE[3:0]:** Package information code

This bitfield indicates the device package type.

**GP device package:**
0000: VFQFPN68 GP
0001: TFBGA100 SMPS GP
0010: LQFP100 GP
0011: LQFP144 GP
0100: UFBGA144 SMPS GP
0111: UFBGA169 SMPS GP
1000: UFBGA176+25 SMPS GP
1001: LQFP176 SMPS GP
1100: TFBGA225 OCTO
1101: TFBGA225 HEXA
1110: WLCSP SMPS GP

**Graphics device package:**
0101: UFBGA144 SMPS GFX
0110: UFBGA169 SMPS GFX
1010: LQFP176 SMPS GFX
1011: UFBGA176+25 SMPS GFX
1100: TFBGA225 OCTO
1101: TFBGA225 HEXA

Bits 15:0 Reserved, must be kept at reset value.

The SBS clock should be enabled first in the RCC_APB4ENR register.
The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST’s customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on www.st.com for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers should re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers should not assume it is certified.

- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.

- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.

- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.

- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.
Table 793. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-Feb-2023</td>
<td>1</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>
| 13-Apr-2023    | 2        | Updated:  
|                |          | – Section 2: Memory and bus architecture  
|                |          | – Section 3: RAMECC monitoring (RAMECC)  
|                |          | – Section 5: Embedded flash memory (FLASH)  
|                |          | – Section 6: Power control (PWR)  
|                |          | – Section 7: Reset and clock control (RCC)  
|                |          | – Section 10: General-purpose I/Os (GPIO)  
|                |          | – Section 43: Basic timers (TIM6/TIM7) |
| 21-Jun-2023    | 3        | Updated:  
|                |          | – Section 2: Memory and bus architecture  
|                |          | – Section 4: System security  
|                |          | – Section 5: Embedded flash memory (FLASH)  
|                |          | – Section 7: Reset and clock control (RCC)  
|                |          | – Section 8: System configuration, boot and security (SBS)  
|                |          | – Section 10: General-purpose I/Os (GPIO)  
|                |          | – Section 66: Debug infrastructure |
| 10-Oct-2023    | 4        | Updated:  
|                |          | – Table 10: Availability of security features  
|                |          | – Figure 1: System architecture  
|                |          | – Table 7: Register boundary addresses  
|                |          | – Table 6: Secure boot process  
|                |          | – Iterated register syntax in Section 10.4: GPIO registers  
|                |          | – Table 85: GPIO register map and reset values |
| 09-Jan-2024    | 5        | Updated:  
|                |          | – Figure 3: Memory map  
|                |          | – Response to tampers  
|                |          | – Section 5.9.22: FLASH RoT status register (FLASH_ROTSR)  
|                |          | – Figure 33: $V_{CORE}$ voltage scaling versus system power modes  
|                |          | – Section 6.8.4: PWR control register 2 (PWR_CSR2)  
|                |          | – Section 10.3.16: High-speed low-voltage mode (HSLV)  
|                |          | – Figure 1004: APB-D CoreSight component topology |
## Updated:

- Read latency in Table 29: FLASH recommended read wait states and programming delays and Section 5.9.1: FLASH access control register (FLASH_ACR)
- IROT_SELECT value in Table 5.4.7: Description of security option bytes
- Section 5.9.22: FLASH RoT status register (FLASH_ROTSR)
- Section 67.1: Unique device ID register (96 bits)
- Section 67.2: Package data register

## Aligned capitalization:

- (S)VOS LOW -> (S)VOS low
- (S)VOS HIGH -> (S)VOS high
- STiROT -> STiRoT

### Table 793. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12-Mar-2024</td>
<td>6</td>
<td>Updated:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Read latency in Table 29: FLASH recommended read wait states and programming delays and Section 5.9.1: FLASH access control register (FLASH_ACR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- IROT_SELECT value in Table 5.4.7: Description of security option bytes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 5.9.22: FLASH RoT status register (FLASH_ROTSR)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 67.1: Unique device ID register (96 bits)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Section 67.2: Package data register</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Aligned capitalization:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- (S)VOS LOW -&gt; (S)VOS low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- (S)VOS HIGH -&gt; (S)VOS high</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- STiROT -&gt; STiRoT</td>
</tr>
</tbody>
</table>
# Index

## A

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC_AWD2CR</td>
<td>1188</td>
</tr>
<tr>
<td>ADC_AWD3CR</td>
<td>1188</td>
</tr>
<tr>
<td>ADC_CALFACT</td>
<td>1190</td>
</tr>
<tr>
<td>ADC_CCR</td>
<td>1193</td>
</tr>
<tr>
<td>ADC_CDR</td>
<td>1196</td>
</tr>
<tr>
<td>ADC_CFGGR</td>
<td>1169</td>
</tr>
<tr>
<td>ADC_CFGGR2</td>
<td>1173</td>
</tr>
<tr>
<td>ADC_CR</td>
<td>1165</td>
</tr>
<tr>
<td>ADC_CSR</td>
<td>1191</td>
</tr>
<tr>
<td>ADC_DIFSEL</td>
<td>1189</td>
</tr>
<tr>
<td>ADC_DR</td>
<td>1183</td>
</tr>
<tr>
<td>ADC_HWCFGR0</td>
<td>1196</td>
</tr>
<tr>
<td>ADC_IER</td>
<td>1163</td>
</tr>
<tr>
<td>ADC_IPDR</td>
<td>1197</td>
</tr>
<tr>
<td>ADC_ISR</td>
<td>1161</td>
</tr>
<tr>
<td>ADC_JDRy</td>
<td>1187</td>
</tr>
<tr>
<td>ADC_JSRQ</td>
<td>1183</td>
</tr>
<tr>
<td>ADC_OFRy</td>
<td>1186</td>
</tr>
<tr>
<td>ADC_OR</td>
<td>1190</td>
</tr>
<tr>
<td>ADC_SIDR</td>
<td>1198</td>
</tr>
<tr>
<td>ADC_SMPR1</td>
<td>1176</td>
</tr>
<tr>
<td>ADC_SMPR2</td>
<td>1176</td>
</tr>
<tr>
<td>ADC_SQR1</td>
<td>1179</td>
</tr>
<tr>
<td>ADC_SQR2</td>
<td>1180</td>
</tr>
<tr>
<td>ADC_SQR3</td>
<td>1181</td>
</tr>
<tr>
<td>ADC_SQR4</td>
<td>1182</td>
</tr>
<tr>
<td>ADC_TR1</td>
<td>1177</td>
</tr>
<tr>
<td>ADC_TR2</td>
<td>1178</td>
</tr>
<tr>
<td>ADC_TR3</td>
<td>1179</td>
</tr>
<tr>
<td>ADC_VERR</td>
<td>1197</td>
</tr>
<tr>
<td>ADF_BSMXOCR</td>
<td>1285</td>
</tr>
<tr>
<td>ADF_CKGR</td>
<td>1282</td>
</tr>
<tr>
<td>ADF_DFLT0ICCR</td>
<td>1288</td>
</tr>
<tr>
<td>ADF_DFLT0CR</td>
<td>1286</td>
</tr>
<tr>
<td>ADF_DFLT0DR</td>
<td>1297</td>
</tr>
<tr>
<td>ADF_DFLT0IER</td>
<td>1291</td>
</tr>
<tr>
<td>ADF_DFLT0ISR</td>
<td>1292</td>
</tr>
<tr>
<td>ADF_DFLT0RSFR</td>
<td>1289</td>
</tr>
<tr>
<td>ADF_DLYOCR</td>
<td>1290</td>
</tr>
<tr>
<td>ADF_GCR</td>
<td>1281</td>
</tr>
<tr>
<td>ADF_SADANLVR</td>
<td>1297</td>
</tr>
<tr>
<td>ADF_SADCFGFR</td>
<td>1295</td>
</tr>
<tr>
<td>ADF_SADCR</td>
<td>1293</td>
</tr>
<tr>
<td>ADF_SADSIDLVR</td>
<td>1296</td>
</tr>
<tr>
<td>ADF_SITF0CR</td>
<td>1284</td>
</tr>
<tr>
<td>AP_BASE</td>
<td>3584</td>
</tr>
<tr>
<td>AP_BD0</td>
<td>3582</td>
</tr>
<tr>
<td>AP_BD1</td>
<td>3583</td>
</tr>
<tr>
<td>AP_BD2</td>
<td>3583</td>
</tr>
<tr>
<td>AP_BD3</td>
<td>3583</td>
</tr>
<tr>
<td>AP_CSW</td>
<td>3580</td>
</tr>
<tr>
<td>AP_DRW</td>
<td>3582</td>
</tr>
<tr>
<td>AP_IDR</td>
<td>3585</td>
</tr>
<tr>
<td>AP_TAR</td>
<td>3582</td>
</tr>
<tr>
<td>AXI_COMP_ID_0</td>
<td>133</td>
</tr>
<tr>
<td>AXI_COMP_ID_1</td>
<td>133</td>
</tr>
<tr>
<td>AXI_COMP_ID_2</td>
<td>134</td>
</tr>
<tr>
<td>AXI_COMP_ID_3</td>
<td>134</td>
</tr>
<tr>
<td>AXI_INIx_FN_MOD</td>
<td>137</td>
</tr>
<tr>
<td>AXI_INIx_FN_MOD2</td>
<td>138</td>
</tr>
<tr>
<td>AXI_INIx_READ_QOS</td>
<td>138</td>
</tr>
<tr>
<td>AXI_INIx_WRITE_QOS</td>
<td>137</td>
</tr>
<tr>
<td>AXI_PERIP_ID_0</td>
<td>131</td>
</tr>
<tr>
<td>AXI_PERIP_ID_1</td>
<td>132</td>
</tr>
<tr>
<td>AXI_PERIP_ID_2</td>
<td>132</td>
</tr>
<tr>
<td>AXI_PERIP_ID_4</td>
<td>131</td>
</tr>
<tr>
<td>AXI_TARG2_FN_MOD_LB</td>
<td>136</td>
</tr>
<tr>
<td>AXI_TARG2_FN_MOD2</td>
<td>136</td>
</tr>
<tr>
<td>AXI_TARGx_FN_MOD</td>
<td>135</td>
</tr>
<tr>
<td>AXI_TARGx_FN_MOD_ISS_BM</td>
<td>135</td>
</tr>
</tbody>
</table>

## C

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEC_CFRG</td>
<td>3548</td>
</tr>
<tr>
<td>CEC_CR</td>
<td>3547</td>
</tr>
<tr>
<td>CEC_IER</td>
<td>3552</td>
</tr>
<tr>
<td>CEC_ISR</td>
<td>3550</td>
</tr>
<tr>
<td>CEC_RXDR</td>
<td>3550</td>
</tr>
<tr>
<td>CEC_TXDR</td>
<td>3550</td>
</tr>
<tr>
<td>CORDIC_CSR</td>
<td>909</td>
</tr>
<tr>
<td>CORDIC_RXDATA</td>
<td>912</td>
</tr>
<tr>
<td>CORDIC_WDATA</td>
<td>911</td>
</tr>
<tr>
<td>CRC_CR</td>
<td>892</td>
</tr>
<tr>
<td>CRC_DR</td>
<td>891</td>
</tr>
<tr>
<td>CRC_IDR</td>
<td>891</td>
</tr>
<tr>
<td>CRC_INIT</td>
<td>893</td>
</tr>
<tr>
<td>CRC_POL</td>
<td>893</td>
</tr>
<tr>
<td>CRS_CFGR</td>
<td>567</td>
</tr>
<tr>
<td>CRS_CR</td>
<td>565</td>
</tr>
<tr>
<td>CRS_ICR</td>
<td>569</td>
</tr>
<tr>
<td>CRS_ISR</td>
<td>568</td>
</tr>
<tr>
<td>CRYP_CFR</td>
<td>1535</td>
</tr>
<tr>
<td>CRYP_CSGCMCCMr</td>
<td>1547</td>
</tr>
<tr>
<td>CRYP_CSGCMxR</td>
<td>1548</td>
</tr>
<tr>
<td>CRYP_DINR</td>
<td>1539</td>
</tr>
<tr>
<td>CRYP_DMCR</td>
<td>1540</td>
</tr>
</tbody>
</table>

---

RM0477 Rev 6 3777/3791
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0477</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRYP_DOUTR</td>
<td>1539</td>
</tr>
<tr>
<td>CRYP_IMSCR</td>
<td>1540</td>
</tr>
<tr>
<td>CRYP_IVOLR</td>
<td>1545</td>
</tr>
<tr>
<td>CRYP_IV0RR</td>
<td>1546</td>
</tr>
<tr>
<td>CRYP_IV1LR</td>
<td>1546</td>
</tr>
<tr>
<td>CRYP_K0LR</td>
<td>1542</td>
</tr>
<tr>
<td>CRYP_K0RR</td>
<td>1543</td>
</tr>
<tr>
<td>CRYP_K1LR</td>
<td>1543</td>
</tr>
<tr>
<td>CRYP_K2LR</td>
<td>1544</td>
</tr>
<tr>
<td>CRYP_K2RR</td>
<td>1544</td>
</tr>
<tr>
<td>CRYP_K3LR</td>
<td>1545</td>
</tr>
<tr>
<td>CRYP_K3RR</td>
<td>1545</td>
</tr>
<tr>
<td>CRYP_MISR</td>
<td>1541</td>
</tr>
<tr>
<td>CRYP_RISR</td>
<td>1541</td>
</tr>
<tr>
<td>CRYP_SR</td>
<td>1537</td>
</tr>
<tr>
<td>CSTF_AUTHSTAT</td>
<td>3620</td>
</tr>
<tr>
<td>CSTF_CIDR0</td>
<td>3623</td>
</tr>
<tr>
<td>CSTF_CIDR1</td>
<td>3624</td>
</tr>
<tr>
<td>CSTF_CIDR2</td>
<td>3624</td>
</tr>
<tr>
<td>CSTF_CIDR3</td>
<td>3624</td>
</tr>
<tr>
<td>CSTF_CLAIMCLR</td>
<td>3618</td>
</tr>
<tr>
<td>CSTF_CLAIMSET</td>
<td>3618</td>
</tr>
<tr>
<td>CSTF_CTRL</td>
<td>3617</td>
</tr>
<tr>
<td>CSTF_DEVID</td>
<td>3620</td>
</tr>
<tr>
<td>CSTF_DEVTYP</td>
<td>3621</td>
</tr>
<tr>
<td>CSTF_LAR</td>
<td>3619</td>
</tr>
<tr>
<td>CSTF_LSR</td>
<td>3619</td>
</tr>
<tr>
<td>CSTF_PIDR0</td>
<td>3622</td>
</tr>
<tr>
<td>CSTF_PIDR1</td>
<td>3622</td>
</tr>
<tr>
<td>CSTF_PIDR2</td>
<td>3622</td>
</tr>
<tr>
<td>CSTF_PIDR3</td>
<td>3623</td>
</tr>
<tr>
<td>CSTF_PIDR4</td>
<td>3621</td>
</tr>
<tr>
<td>CSTF_PRIORITY</td>
<td>3617</td>
</tr>
<tr>
<td>CTI_APPCLEAR</td>
<td>3601</td>
</tr>
<tr>
<td>CTI_APPPULSE</td>
<td>3602</td>
</tr>
<tr>
<td>CTI_APPSET</td>
<td>3601</td>
</tr>
<tr>
<td>CTI_AUTHSTAT</td>
<td>3608</td>
</tr>
<tr>
<td>CTI_CHINSTS</td>
<td>3604</td>
</tr>
<tr>
<td>CTI_CHOUTSTS</td>
<td>3605</td>
</tr>
<tr>
<td>CTI_CIDR0</td>
<td>3611</td>
</tr>
<tr>
<td>CTI_CIDR1</td>
<td>3611</td>
</tr>
<tr>
<td>CTI_CIDR2</td>
<td>3612</td>
</tr>
<tr>
<td>CTI_CIDR3</td>
<td>3612</td>
</tr>
<tr>
<td>CTI_CLAIMCLR</td>
<td>3606</td>
</tr>
<tr>
<td>CTI_CLAIMSET</td>
<td>3606</td>
</tr>
<tr>
<td>CTI_CONTROL</td>
<td>3600</td>
</tr>
<tr>
<td>CTI_DEVID</td>
<td>3608</td>
</tr>
<tr>
<td>CTI_DEVTYP</td>
<td>3609</td>
</tr>
<tr>
<td>CTI_GATE</td>
<td>3605</td>
</tr>
<tr>
<td>CTI_INENx</td>
<td>3602</td>
</tr>
<tr>
<td>CTI_INTACK</td>
<td>3600</td>
</tr>
<tr>
<td>CTI_LAR</td>
<td>3607</td>
</tr>
<tr>
<td>CTI_LSR</td>
<td>3607</td>
</tr>
<tr>
<td>CTI_OUTENx</td>
<td>3603</td>
</tr>
<tr>
<td>CTI_PIDR0</td>
<td>3610</td>
</tr>
<tr>
<td>CTI_PIDR1</td>
<td>3610</td>
</tr>
<tr>
<td>CTI_PIDR2</td>
<td>3610</td>
</tr>
<tr>
<td>CTI_PIDR3</td>
<td>3611</td>
</tr>
<tr>
<td>CTI_PIDR4</td>
<td>3609</td>
</tr>
<tr>
<td>CTI_TRGISTS</td>
<td>3603</td>
</tr>
<tr>
<td>CTI_TRGOSTS</td>
<td>3604</td>
</tr>
<tr>
<td>CRYP_IV0RR</td>
<td>1546</td>
</tr>
<tr>
<td>CRYP_IV1LR</td>
<td>1546</td>
</tr>
<tr>
<td>CRYP_K0LR</td>
<td>1542</td>
</tr>
<tr>
<td>CRYP_K0RR</td>
<td>1543</td>
</tr>
<tr>
<td>CRYP_K1LR</td>
<td>1543</td>
</tr>
<tr>
<td>CRYP_K2LR</td>
<td>1544</td>
</tr>
<tr>
<td>CRYP_K2RR</td>
<td>1544</td>
</tr>
<tr>
<td>CRYP_K3LR</td>
<td>1545</td>
</tr>
<tr>
<td>CRYP_K3RR</td>
<td>1545</td>
</tr>
<tr>
<td>INDEX</td>
<td></td>
</tr>
<tr>
<td>CRYP_MISR</td>
<td>1541</td>
</tr>
<tr>
<td>CRYP_RISR</td>
<td>1541</td>
</tr>
<tr>
<td>CRYP_SR</td>
<td>1537</td>
</tr>
<tr>
<td>CSTF_AUTHSTAT</td>
<td>3620</td>
</tr>
<tr>
<td>CSTF_CIDR0</td>
<td>3623</td>
</tr>
<tr>
<td>CSTF_CIDR1</td>
<td>3624</td>
</tr>
<tr>
<td>CSTF_CIDR2</td>
<td>3624</td>
</tr>
<tr>
<td>CSTF_CIDR3</td>
<td>3624</td>
</tr>
<tr>
<td>CSTF_CLAIMCLR</td>
<td>3618</td>
</tr>
<tr>
<td>CSTF_CLAIMSET</td>
<td>3618</td>
</tr>
<tr>
<td>CSTF_CTRL</td>
<td>3617</td>
</tr>
<tr>
<td>CSTF_DEVID</td>
<td>3620</td>
</tr>
<tr>
<td>CSTF_DEVTYP</td>
<td>3621</td>
</tr>
<tr>
<td>CSTF_LAR</td>
<td>3619</td>
</tr>
<tr>
<td>CSTF_LSR</td>
<td>3619</td>
</tr>
<tr>
<td>CSTF_PIDR0</td>
<td>3622</td>
</tr>
<tr>
<td>CSTF_PIDR1</td>
<td>3622</td>
</tr>
<tr>
<td>CSTF_PIDR2</td>
<td>3622</td>
</tr>
<tr>
<td>CSTF_PIDR3</td>
<td>3623</td>
</tr>
<tr>
<td>CSTF_PIDR4</td>
<td>3621</td>
</tr>
<tr>
<td>CSTF_PRIORITY</td>
<td>3617</td>
</tr>
<tr>
<td>CTI_APPCLEAR</td>
<td>3601</td>
</tr>
<tr>
<td>CTI_APPPULSE</td>
<td>3602</td>
</tr>
<tr>
<td>CTI_APPSET</td>
<td>3601</td>
</tr>
<tr>
<td>CTI_AUTHSTAT</td>
<td>3608</td>
</tr>
<tr>
<td>CTI_CHINSTS</td>
<td>3604</td>
</tr>
<tr>
<td>CTI_CHOUTSTS</td>
<td>3605</td>
</tr>
<tr>
<td>CTI_CIDR0</td>
<td>3611</td>
</tr>
<tr>
<td>CTI_CIDR1</td>
<td>3611</td>
</tr>
<tr>
<td>CTI_CIDR2</td>
<td>3612</td>
</tr>
<tr>
<td>CTI_CIDR3</td>
<td>3612</td>
</tr>
<tr>
<td>CTI_CLAIMCLR</td>
<td>3606</td>
</tr>
<tr>
<td>CTI_CLAIMSET</td>
<td>3606</td>
</tr>
<tr>
<td>CTI_CONTROL</td>
<td>3600</td>
</tr>
<tr>
<td>CTI_DEVID</td>
<td>3608</td>
</tr>
<tr>
<td>CTI_DEVTYP</td>
<td>3609</td>
</tr>
<tr>
<td>CTI_GATE</td>
<td>3605</td>
</tr>
<tr>
<td>CTI_INENx</td>
<td>3602</td>
</tr>
</tbody>
</table>

**D**

<table>
<thead>
<tr>
<th>Index</th>
<th>RM0477</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBGMCU_AHB1FZR</td>
<td>3684</td>
</tr>
<tr>
<td>DBGMCU_AHB5FZR</td>
<td>3682</td>
</tr>
<tr>
<td>DBGMCU_APB1FZR</td>
<td>3685</td>
</tr>
<tr>
<td>DBGMCU_APB2FZ</td>
<td>3687</td>
</tr>
<tr>
<td>DBGMCU_APB4FZR</td>
<td>3687</td>
</tr>
<tr>
<td>DBGMCU_CIDR0</td>
<td>3692</td>
</tr>
<tr>
<td>DBGMCU_CIDR1</td>
<td>3693</td>
</tr>
<tr>
<td>DBGMCU_CIDR2</td>
<td>3693</td>
</tr>
<tr>
<td>DBGMCU_CIDR3</td>
<td>3693</td>
</tr>
<tr>
<td>DBGMCU_CR</td>
<td>3681</td>
</tr>
<tr>
<td>DBGMCU_DBG_AUTH_ACK</td>
<td>3690</td>
</tr>
<tr>
<td>DBGMCU_DBG_AUTH_DEVICE</td>
<td>3689</td>
</tr>
<tr>
<td>DBGMCU_DBG_AUTH_HOST</td>
<td>3689</td>
</tr>
<tr>
<td>DBGMCU_IDCODE</td>
<td>3681</td>
</tr>
<tr>
<td>DBGMCU_PIDR0</td>
<td>3691</td>
</tr>
<tr>
<td>DBGMCU_PIDR1</td>
<td>3691</td>
</tr>
<tr>
<td>DBGMCU_PIDR2</td>
<td>3691</td>
</tr>
<tr>
<td>DBGMCU_PIDR3</td>
<td>3692</td>
</tr>
<tr>
<td>DBGMCU_PIDR4</td>
<td>3690</td>
</tr>
<tr>
<td>DBGMCU_SR</td>
<td>3688</td>
</tr>
<tr>
<td>DCMIPP_CMCR</td>
<td>1339</td>
</tr>
<tr>
<td>DCMIPP_CMFCR</td>
<td>1344</td>
</tr>
<tr>
<td>DCMIPP_CMFRCR</td>
<td>1340</td>
</tr>
<tr>
<td>DCMIPP_CMIER</td>
<td>1340</td>
</tr>
<tr>
<td>DCMIPP_CMSR1</td>
<td>1341</td>
</tr>
<tr>
<td>DCMIPP_CMSR2</td>
<td>1342</td>
</tr>
<tr>
<td>DCMIPP_IPCxR1</td>
<td>1333</td>
</tr>
<tr>
<td>DCMIPP_IPCxR2</td>
<td>1334</td>
</tr>
<tr>
<td>DCMIPP_IPCxR3</td>
<td>1334</td>
</tr>
<tr>
<td>DCMIPP_IPGR1</td>
<td>1331</td>
</tr>
<tr>
<td>DCMIPP_IPGR2</td>
<td>1332</td>
</tr>
<tr>
<td>DCMIPP_IPGR3</td>
<td>1332</td>
</tr>
<tr>
<td>DCMIPP_IPGR8</td>
<td>1332</td>
</tr>
<tr>
<td>DCMIPP_P0CFCTR</td>
<td>1353</td>
</tr>
<tr>
<td>DCMIPP_P0CPCCR</td>
<td>1355</td>
</tr>
<tr>
<td>DCMIPP_P0CPPM0AR1</td>
<td>1356</td>
</tr>
<tr>
<td>DCMIPP_P0CPPM0AR2</td>
<td>1356</td>
</tr>
<tr>
<td>DCMIPP_P0SCSCTR</td>
<td>1353</td>
</tr>
</tbody>
</table>
DCMIPP_P0SCSZR ............... 1354
DCMIPP_P0CCNT ............... 1347
DCMIPP_P0CLMTR ............... 1348
DCMIPP_P0CFR ............... 1352
DCMIPP_P0CTR ............... 1345
DCMIPP_P0FCTR ............... 1345
DCMIPP_P0IER ............... 1350
DCMIPP_P0PCCR ............... 1348
DCMIPP_P0PPM0AR1 ............ 1349
DCMIPP_P0PPM0AR2 ............ 1350
DCMIPP_P0SCSTR ............... 1346
DCMIPP_P0SCSZR ............... 1346
DCMIPP_P0SR ............... 1351
DCMIPP_P0CR ............... 1355
DCMIPP_P0FSCR ............... 1354
DCMIPP_P0GCTR ............... 1346
DCMIPP_P0GCTR ............... 1346
DCMIPP_P0OSR ............... 1351
DCMIPP_P0OR ............... 1351
DCMIPP_P0PRF ............... 1338
DCMIPP_P0PRFCCR ............. 1345
DCMIPP_P0SCSZR ............... 1346
DTS_CFGR1 ............... 1211
DTS_CFGR1 ............... 1211
DTS_DR ............... 1214
DTS_ICIFR ............... 1217
DTS_ITENR ............... 1216
DTS_IWR ............... 1214
DTS_OR ............... 1218
DTS_RAMPVAL ............... 1213
DTS_SR ............... 1215
DTS_TOVALR ............... 1213
DCMIPP_P0SCSZR ............... 1346
DTS_CFGR1 ............... 1211
DTS_DR ............... 1214
DTS_ICIFR ............... 1217
DTS_ITENR ............... 1216
DTS_IWR ............... 1214
DTS_OR ............... 1218
DTS_RAMPVAL ............... 1213
DTS_SR ............... 1215
DTS_TOVALR ............... 1213
ETF_AUTHSTAT ............... 3641
ETF_BUFWM ............... 3636
ETF_CBURLVL ............... 3635
ETF_CIDR0 ............... 3645
ETF_CIDR1 ............... 3645
ETF_CIDR2 ............... 3645
ETF_CIDR3 ............... 3646
ETFCLAIMCLR ............... 3640
ETFCLAIMSET ............... 3639
ETF_CTL ............... 3633
ETF_DEVID ............... 3642
ETF_DEVTYPE ............... 3642
ETF_FFCR ............... 3637
ETF_FFSR ............... 3636
ETF_LAR ............... 3640
ETF_LBUFLVL ............... 3635
ETF_LSR ............... 3641
ETF_MODE ............... 3634
ETF_PIDR0 ............... 3643
ETF_PIDR1 ............... 3643
ETF_PIDR2 ............... 3644
ETF_PIDR3 ............... 3644
ETF_PIDR4 ............... 3643
ETF_PSCR ............... 3639
ETF_RRD ............... 3631
ETF_RRP ............... 3632
ETF_RSZ ............... 3630
ETF_RWD ............... 3634
ETF_RWP ............... 3632
ETF_STS ............... 3630
ETF_TRG ............... 3633
ETH_DMACCARXBR ............ 3422
ETH_DMACCARXDR ............ 3421
ETH_DMACCATXBR ............ 3422
ETH_DMACCATXDR ............ 3421
ETH_DMACC ............... 3409
ETH_DMACIER ............... 3416
ETH_DMACMFCR ............... 3425
ETH_DMACRXCR ............... 3411
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0477</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH_DMCRXDXLAR</td>
<td>ETH_MACPCSR</td>
</tr>
<tr>
<td>ETH_DMCRXDTPLR</td>
<td>ETH_MACPFR</td>
</tr>
<tr>
<td>ETH_DMCRXIVWR</td>
<td>ETH_MACPOCR</td>
</tr>
<tr>
<td>ETH_DMCRXRLR</td>
<td>ETH_MACPPSCR</td>
</tr>
<tr>
<td>ETH_DMCRS</td>
<td>ETH_MACPPSR</td>
</tr>
<tr>
<td>ETH_DMCTR</td>
<td>ETH_MACPPSTTNR</td>
</tr>
<tr>
<td>ETH_DMCTXDLR</td>
<td>ETH_MACPPSTTSR</td>
</tr>
<tr>
<td>ETH_DMCTXTPR</td>
<td>ETH_MACPPSWR</td>
</tr>
<tr>
<td>ETH_DMCTXRLR</td>
<td>ETH_MACQTXFCR</td>
</tr>
<tr>
<td>ETH_DMDSR</td>
<td>ETH_MACRKPSFR</td>
</tr>
<tr>
<td>ETH_DMAISR</td>
<td>ETH_MACRXFCR</td>
</tr>
<tr>
<td>ETH_DMAMR</td>
<td>ETH_MACRXTXSR</td>
</tr>
<tr>
<td>ETH_DMSBMR</td>
<td>ETH_MACSPI0R</td>
</tr>
<tr>
<td>ETH_MACIUSTCR</td>
<td>ETH_MACSPI1R</td>
</tr>
<tr>
<td>ETH_MACAOHR</td>
<td>ETH_MACSPI2R</td>
</tr>
<tr>
<td>ETH_MACACR</td>
<td>ETH_MACSSIR</td>
</tr>
<tr>
<td>ETH_MACARP</td>
<td>ETH_MACSTNAR</td>
</tr>
<tr>
<td>ETH_MACATSNR</td>
<td>ETH_MACSTNUR</td>
</tr>
<tr>
<td>ETH_MACATSSR</td>
<td>ETH_MACSTS</td>
</tr>
<tr>
<td>ETH_MACAXHR</td>
<td>ETH_MACSTSUR</td>
</tr>
<tr>
<td>ETH_MACAXLR</td>
<td>ETH_MACTSAR</td>
</tr>
<tr>
<td>ETH_MACCR</td>
<td>ETH_MACTS</td>
</tr>
<tr>
<td>ETH_MACCSSRWCR</td>
<td>ETH_MACTSEACR</td>
</tr>
<tr>
<td>ETH_MACD</td>
<td>ETH_MACSECNR</td>
</tr>
<tr>
<td>ETH_MACECR</td>
<td>ETH_MACSIACR</td>
</tr>
<tr>
<td>ETH_MACH0R</td>
<td>ETH_MACSIICR</td>
</tr>
<tr>
<td>ETH_MACHT1R</td>
<td>ETH_MACSISR</td>
</tr>
<tr>
<td>ETH_MACHWF0R</td>
<td>ETH_MACSSSR</td>
</tr>
<tr>
<td>ETH_MACHWF1R</td>
<td>ETH_MACTSSSR</td>
</tr>
<tr>
<td>ETH_MACHWF2R</td>
<td>ETH_MACVHTR</td>
</tr>
<tr>
<td>ETH_MACHWF3R</td>
<td>ETH_MACVIR</td>
</tr>
<tr>
<td>ETH_MACIER</td>
<td>ETH_MACVR</td>
</tr>
<tr>
<td>ETH_MACISR</td>
<td>ETH_MACVTR</td>
</tr>
<tr>
<td>ETH_MACIVIR</td>
<td>ETH_MACWTR</td>
</tr>
<tr>
<td>ETH_MACL3A00R</td>
<td>ETH_MMC_CONTROL</td>
</tr>
<tr>
<td>ETH_MACL3A01R</td>
<td>ETH_MMC_RX_INTERRUPT</td>
</tr>
<tr>
<td>ETH_MACL3A10R</td>
<td>ETH_MMC_RX_INTERRUPT_MASK</td>
</tr>
<tr>
<td>ETH_MACL3A11R</td>
<td>ETH_MMC_TX_INTERRUPT</td>
</tr>
<tr>
<td>ETH_MACL3A20R</td>
<td>ETH_MMC_TX_INTERRUPT_MASK</td>
</tr>
<tr>
<td>ETH_MACL3A21R</td>
<td>ETH_MTLISR</td>
</tr>
<tr>
<td>ETH_MACL3A30R</td>
<td>ETH_MTLOMR</td>
</tr>
<tr>
<td>ETH_MACL3A31R</td>
<td>ETH_MTLQICSR</td>
</tr>
<tr>
<td>ETH_MACL3L4CR</td>
<td>ETH_MTLRXQDR</td>
</tr>
<tr>
<td>ETH_MACL3L4CR</td>
<td>ETH_MTLRXQMPOCR</td>
</tr>
<tr>
<td>ETH_MACL4A0R</td>
<td>ETH_MTLRXQDMR</td>
</tr>
<tr>
<td>ETH_MACL4AR</td>
<td>ETH_MTLRXQOMR</td>
</tr>
<tr>
<td>ETH_MACLCSR</td>
<td>ETH_MTLRXQOMR</td>
</tr>
<tr>
<td>ETH_MACLETR</td>
<td>ETH_MTLTXQUR</td>
</tr>
<tr>
<td>ETH_MACLMIR</td>
<td>ETH_MTLTXQUR</td>
</tr>
<tr>
<td>ETH_MACLTCR</td>
<td>ETH_RX_ALIGNMENT_ERROR_PACKETS</td>
</tr>
<tr>
<td>ETH_MACMDOAR</td>
<td>ETH_RX_CRC_ERROR_PACKETS</td>
</tr>
<tr>
<td>ETH_MACMDIODR</td>
<td>ETH_RX_LPI_TRAN_CNTR</td>
</tr>
</tbody>
</table>
**RM0477 Index**

<table>
<thead>
<tr>
<th>ETH_RX_LPI_USEC_CNTR</th>
<th>3494</th>
</tr>
</thead>
<tbody>
<tr>
<td>ETH_RX_UNICAST_PACKETS_GOOD</td>
<td>3493</td>
</tr>
<tr>
<td>ETH_TX_LPI_TRAN_CNTR</td>
<td>3494</td>
</tr>
<tr>
<td>ETH_TX_LPI_USEC_CNTR</td>
<td>3493</td>
</tr>
<tr>
<td>ETH_TX_MULTIPLE_COLLISION_</td>
<td>3491</td>
</tr>
<tr>
<td>GOOD_PACKETS</td>
<td></td>
</tr>
<tr>
<td>ETH_TX_PACKET_COUNT_GOOD</td>
<td>3492</td>
</tr>
<tr>
<td>ETH_TX_SINGLE_COLLISION_GOOD_PACKETS</td>
<td>3491</td>
</tr>
<tr>
<td>ETS</td>
<td></td>
</tr>
<tr>
<td>EXTI_EMR1</td>
<td>882</td>
</tr>
<tr>
<td>EXTI_EMRS</td>
<td>883</td>
</tr>
<tr>
<td>EXTI_FTSR1</td>
<td>877</td>
</tr>
<tr>
<td>EXTI_FTSR2</td>
<td>879</td>
</tr>
<tr>
<td>EXTI_IMR1</td>
<td>881</td>
</tr>
<tr>
<td>EXTI_IMRS</td>
<td>883</td>
</tr>
<tr>
<td>EXTI_PR1</td>
<td>884</td>
</tr>
<tr>
<td>EXTI_PR2</td>
<td>884</td>
</tr>
<tr>
<td>EXTI_RTSR1</td>
<td>877</td>
</tr>
<tr>
<td>EXTI_RTSR2</td>
<td>878</td>
</tr>
<tr>
<td>EXTI_SWIER1</td>
<td>878</td>
</tr>
<tr>
<td>EXTI_SWIER2</td>
<td>880</td>
</tr>
<tr>
<td>FDCAN_CCCR</td>
<td>2864</td>
</tr>
<tr>
<td>FDCAN_CKDIV</td>
<td>2889</td>
</tr>
<tr>
<td>FDCAN_CREL</td>
<td>2861</td>
</tr>
<tr>
<td>FDCAN_DBTP</td>
<td>2861</td>
</tr>
<tr>
<td>FDCAN_ECR</td>
<td>2869</td>
</tr>
<tr>
<td>FDCAN_ENDN</td>
<td>2861</td>
</tr>
<tr>
<td>FDCAN_HFMS</td>
<td>2880</td>
</tr>
<tr>
<td>FDCAN_IE</td>
<td>2875</td>
</tr>
<tr>
<td>FDCAN_IE</td>
<td>2875</td>
</tr>
<tr>
<td>FDCAN_IE</td>
<td>2878</td>
</tr>
<tr>
<td>FDCAN_ILS</td>
<td>2877</td>
</tr>
<tr>
<td>FDCAN_JR</td>
<td>2872</td>
</tr>
<tr>
<td>FDCAN_NBTP</td>
<td>2865</td>
</tr>
<tr>
<td>FDCAN_PSR</td>
<td>2870</td>
</tr>
<tr>
<td>FDCAN_RWD</td>
<td>2863</td>
</tr>
<tr>
<td>FDCAN_RXFOA</td>
<td>2882</td>
</tr>
<tr>
<td>FDCAN_RXFOE</td>
<td>2881</td>
</tr>
<tr>
<td>FDCAN_RXF1A</td>
<td>2883</td>
</tr>
<tr>
<td>FDCAN_RXF1S</td>
<td>2882</td>
</tr>
<tr>
<td>FDCAN_RXGC</td>
<td>2878</td>
</tr>
<tr>
<td>FDCAN_TDCR</td>
<td>2872</td>
</tr>
<tr>
<td>FDCAN_TEST</td>
<td>2862</td>
</tr>
<tr>
<td>FMC_BCRx</td>
<td>954</td>
</tr>
<tr>
<td>FMC_BTRx</td>
<td>958</td>
</tr>
<tr>
<td>FMC_BWRx</td>
<td>961</td>
</tr>
<tr>
<td>FMC_ECCR</td>
<td>974</td>
</tr>
<tr>
<td>FMC_PATT</td>
<td>972</td>
</tr>
<tr>
<td>FMC_PR</td>
<td>969</td>
</tr>
<tr>
<td>Index</td>
<td>RM0477</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
</tr>
<tr>
<td>FMC_PMEM</td>
<td>971</td>
</tr>
<tr>
<td>FMC_SDCMR</td>
<td>989</td>
</tr>
<tr>
<td>FMC_SDCRx</td>
<td>986</td>
</tr>
<tr>
<td>FMC_SDRTR</td>
<td>990</td>
</tr>
<tr>
<td>FMC_SDRS</td>
<td>992</td>
</tr>
<tr>
<td>FMC_SDTRx</td>
<td>987</td>
</tr>
<tr>
<td>FMC_SR</td>
<td>970</td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>GFXMMU_BxCR</td>
<td>770</td>
</tr>
<tr>
<td>GFXMMU_CR</td>
<td>766</td>
</tr>
<tr>
<td>GFXMMU_DAR</td>
<td>769</td>
</tr>
<tr>
<td>GFXMMU_DVR</td>
<td>769</td>
</tr>
<tr>
<td>GFXMMU_FCR</td>
<td>768</td>
</tr>
<tr>
<td>GFXMMU_LUTxH</td>
<td>771</td>
</tr>
<tr>
<td>GFXMMU_LUTxL</td>
<td>770</td>
</tr>
<tr>
<td>GFXMMU_SR</td>
<td>768</td>
</tr>
<tr>
<td>GFXTIM_AFC1R</td>
<td>854</td>
</tr>
<tr>
<td>GFXTIM_AFCR</td>
<td>853</td>
</tr>
<tr>
<td>GFXTIM_ALCC1R</td>
<td>854</td>
</tr>
<tr>
<td>GFXTIM_ALCC2R</td>
<td>854</td>
</tr>
<tr>
<td>GFXTIM_ALCR</td>
<td>853</td>
</tr>
<tr>
<td>GFXTIM_ATR</td>
<td>852</td>
</tr>
<tr>
<td>GFXTIM_DBGCR</td>
<td>838</td>
</tr>
<tr>
<td>GFXTIM_CR</td>
<td>837</td>
</tr>
<tr>
<td>GFXTIM_EVCR</td>
<td>842</td>
</tr>
<tr>
<td>GFXTIM_EVSR</td>
<td>842</td>
</tr>
<tr>
<td>GFXTIM_FCCRR</td>
<td>852</td>
</tr>
<tr>
<td>GFXTIM_ICR</td>
<td>847</td>
</tr>
<tr>
<td>GFXTIM_IER</td>
<td>849</td>
</tr>
<tr>
<td>GFXTIM_ISR</td>
<td>846</td>
</tr>
<tr>
<td>GFXTIM_LCCRR</td>
<td>852</td>
</tr>
<tr>
<td>GFXTIM_LCDC1R</td>
<td>855</td>
</tr>
<tr>
<td>GFXTIM_LCDC1RR</td>
<td>855</td>
</tr>
<tr>
<td>GFXTIM_LCDC2R</td>
<td>856</td>
</tr>
<tr>
<td>GFXTIM_MISR</td>
<td>856</td>
</tr>
<tr>
<td>GFXTIM_TCR</td>
<td>840</td>
</tr>
<tr>
<td>GFXTIM_TDR</td>
<td>841</td>
</tr>
<tr>
<td>GFXTIM_TSR</td>
<td>851</td>
</tr>
<tr>
<td>GFXTIM_WDGCR</td>
<td>856</td>
</tr>
<tr>
<td>GFXTIM_WDGPAR</td>
<td>857</td>
</tr>
<tr>
<td>GFXTIM_WDGRR</td>
<td>857</td>
</tr>
<tr>
<td>GFXTIM_WDGTCTR</td>
<td>844</td>
</tr>
<tr>
<td>GPDMA_CxBR1</td>
<td>666-663</td>
</tr>
<tr>
<td>GPDMA_CxBR2</td>
<td>669</td>
</tr>
<tr>
<td>GPDMA_CxCR</td>
<td>653</td>
</tr>
<tr>
<td>GPDMA_CxDAR</td>
<td>667</td>
</tr>
<tr>
<td>GPDMA_CxFCR</td>
<td>650</td>
</tr>
<tr>
<td>GPDMA_CxLBAR</td>
<td>650</td>
</tr>
<tr>
<td>GPDMA_CxLLR</td>
<td>670, 672</td>
</tr>
<tr>
<td>GPDMA_CxSAR</td>
<td>666</td>
</tr>
<tr>
<td>GPDMA_CxSR</td>
<td>651</td>
</tr>
<tr>
<td>GPDMA_CxTR1</td>
<td>655</td>
</tr>
<tr>
<td>GPDMA_CxTR2</td>
<td>658</td>
</tr>
<tr>
<td>GPDMA_CxTR3</td>
<td>668</td>
</tr>
<tr>
<td>GPDMA_MISR</td>
<td>649</td>
</tr>
<tr>
<td>GPDMA_PRIVCFG</td>
<td>648</td>
</tr>
<tr>
<td>GPDMA_RCFGLOCKR</td>
<td>649</td>
</tr>
<tr>
<td>GPIOx_AFRH</td>
<td>586</td>
</tr>
<tr>
<td>GPIOx_AFRL</td>
<td>585</td>
</tr>
<tr>
<td>GPIOx_BRR</td>
<td>587</td>
</tr>
<tr>
<td>GPIOx_BSR</td>
<td>583</td>
</tr>
<tr>
<td>GPIOx_IDR</td>
<td>582</td>
</tr>
<tr>
<td>GPIOx_LCKR</td>
<td>584</td>
</tr>
<tr>
<td>GPIOx_MODER</td>
<td>580</td>
</tr>
<tr>
<td>GPIOx_ODR</td>
<td>583</td>
</tr>
<tr>
<td>GPIOx_OSPEDR</td>
<td>581</td>
</tr>
<tr>
<td>GPIOx_OTYPER</td>
<td>581</td>
</tr>
<tr>
<td>GPIOx_PUPDR</td>
<td>582</td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>HASH_CR</td>
<td>1562</td>
</tr>
<tr>
<td>HASH_CSRx</td>
<td>1570</td>
</tr>
<tr>
<td>HASH_DIN</td>
<td>1564</td>
</tr>
<tr>
<td>HASH_HRAx</td>
<td>1567</td>
</tr>
<tr>
<td>HASH_HRx</td>
<td>1567</td>
</tr>
<tr>
<td>HASH_IMR</td>
<td>1568</td>
</tr>
<tr>
<td>HASH_SR</td>
<td>1568</td>
</tr>
<tr>
<td>HASH_STR</td>
<td>1565</td>
</tr>
<tr>
<td>HPDMA_CxBR1</td>
<td>745-746</td>
</tr>
<tr>
<td>HPDMA_CxBR2</td>
<td>752</td>
</tr>
<tr>
<td>HPDMA_CxCR</td>
<td>735</td>
</tr>
<tr>
<td>HPDMA_CxDAR</td>
<td>750</td>
</tr>
<tr>
<td>HPDMA_CxFCR</td>
<td>733</td>
</tr>
<tr>
<td>HPDMA_CxLBAR</td>
<td>732</td>
</tr>
<tr>
<td>HPDMA_CxLLR</td>
<td>753, 755</td>
</tr>
<tr>
<td>HPDMA_CxSAR</td>
<td>749</td>
</tr>
<tr>
<td>HPDMA_CxSR</td>
<td>734</td>
</tr>
<tr>
<td>HPDMA_CxTR1</td>
<td>738</td>
</tr>
<tr>
<td>HPDMA_CxTR2</td>
<td>741</td>
</tr>
<tr>
<td>HPDMA_CxTR3</td>
<td>751</td>
</tr>
<tr>
<td>HPDMA_MISR</td>
<td>731</td>
</tr>
<tr>
<td>HPDMA_PRIVCFG</td>
<td>730</td>
</tr>
<tr>
<td>HPDMA_RCFGLOCKR</td>
<td>731</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>I2C_CR1</td>
<td>2283</td>
</tr>
<tr>
<td>I2C_CR2</td>
<td>2286</td>
</tr>
<tr>
<td>I2C_ICR</td>
<td>2294</td>
</tr>
<tr>
<td>I2C_ISR</td>
<td>2291</td>
</tr>
<tr>
<td>I2C_OAR1</td>
<td>2288</td>
</tr>
<tr>
<td>I2C_OAR2</td>
<td>2288</td>
</tr>
</tbody>
</table>

3782/3791  RM0477 Rev 6
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I2C_PECR</td>
<td></td>
<td>2295</td>
</tr>
<tr>
<td>I2C_RXDR</td>
<td></td>
<td>2295</td>
</tr>
<tr>
<td>I2C_TIMEOUTR</td>
<td></td>
<td>2290</td>
</tr>
<tr>
<td>I2C_TIMINGR</td>
<td></td>
<td>2289</td>
</tr>
<tr>
<td>I2C_TXDR</td>
<td></td>
<td>2296</td>
</tr>
<tr>
<td>I3C_BCR</td>
<td></td>
<td>2397</td>
</tr>
<tr>
<td>I3C_CEVFR</td>
<td></td>
<td>2385</td>
</tr>
<tr>
<td>I3C_CFGFR</td>
<td></td>
<td>2364</td>
</tr>
<tr>
<td>I3C_CR</td>
<td></td>
<td>2360, 2362</td>
</tr>
<tr>
<td>I3C_DCR</td>
<td></td>
<td>2398</td>
</tr>
<tr>
<td>I3C_DEVR0</td>
<td></td>
<td>2387</td>
</tr>
<tr>
<td>I3C_DEVRx</td>
<td></td>
<td>2389</td>
</tr>
<tr>
<td>I3C_EPIIDR</td>
<td></td>
<td>2403</td>
</tr>
<tr>
<td>I3C_EVT</td>
<td></td>
<td>2379</td>
</tr>
<tr>
<td>I3C_GETCAPR</td>
<td></td>
<td>2399</td>
</tr>
<tr>
<td>I3C_GETMXDSR</td>
<td></td>
<td>2401</td>
</tr>
<tr>
<td>I3C_IBIDR</td>
<td></td>
<td>2373</td>
</tr>
<tr>
<td>I3C_IER</td>
<td></td>
<td>2383</td>
</tr>
<tr>
<td>I3C_MAXLR</td>
<td></td>
<td>2391</td>
</tr>
<tr>
<td>I3C_MAXWLR</td>
<td></td>
<td>2392</td>
</tr>
<tr>
<td>I3C_RDR</td>
<td></td>
<td>2369</td>
</tr>
<tr>
<td>I3C_RDWR</td>
<td></td>
<td>2369</td>
</tr>
<tr>
<td>I3C_RMR</td>
<td></td>
<td>2378</td>
</tr>
<tr>
<td>I3C_SER</td>
<td></td>
<td>2376</td>
</tr>
<tr>
<td>I3C_SR</td>
<td></td>
<td>2375</td>
</tr>
<tr>
<td>I3C_TDR</td>
<td></td>
<td>2370</td>
</tr>
<tr>
<td>I3C_TDWR</td>
<td></td>
<td>2371</td>
</tr>
<tr>
<td>I3C_TGTDR</td>
<td></td>
<td>2374</td>
</tr>
<tr>
<td>I3C_TIMINGR0</td>
<td></td>
<td>2393</td>
</tr>
<tr>
<td>I3C_TIMINGR1</td>
<td></td>
<td>2394</td>
</tr>
<tr>
<td>I3C_TIMINGR2</td>
<td></td>
<td>2396</td>
</tr>
<tr>
<td>I3C_CEVR</td>
<td></td>
<td>2385</td>
</tr>
<tr>
<td>ICACHE_CR</td>
<td></td>
<td>819</td>
</tr>
<tr>
<td>ICACHE_FCR</td>
<td></td>
<td>821</td>
</tr>
<tr>
<td>ICACHE_HMONR</td>
<td></td>
<td>822</td>
</tr>
<tr>
<td>ICACHE_IER</td>
<td></td>
<td>821</td>
</tr>
<tr>
<td>ICACHE_MMONR</td>
<td></td>
<td>822</td>
</tr>
<tr>
<td>ICACHE_SR</td>
<td></td>
<td>820</td>
</tr>
<tr>
<td>IWDG_EWCR</td>
<td></td>
<td>2140</td>
</tr>
<tr>
<td>IWDG_KR</td>
<td></td>
<td>2137</td>
</tr>
<tr>
<td>IWDG_PR</td>
<td></td>
<td>2137</td>
</tr>
<tr>
<td>IWDG_RLR</td>
<td></td>
<td>2138</td>
</tr>
<tr>
<td>IWDG_SR</td>
<td></td>
<td>2138</td>
</tr>
<tr>
<td>IWDG_WINR</td>
<td></td>
<td>2140</td>
</tr>
<tr>
<td>JPEG_CONFRx</td>
<td></td>
<td>1416</td>
</tr>
<tr>
<td>JPEG_CR</td>
<td></td>
<td>1417</td>
</tr>
<tr>
<td>JPEG_DHTMEMx</td>
<td></td>
<td>1424</td>
</tr>
<tr>
<td>JPEG_DIR</td>
<td></td>
<td>1420</td>
</tr>
<tr>
<td>JPEG_DOR</td>
<td></td>
<td>1420</td>
</tr>
<tr>
<td>JPEG_HUFFBASEx</td>
<td></td>
<td>1422</td>
</tr>
<tr>
<td>JPEG_HUFFENC_ACx_y</td>
<td></td>
<td>1424</td>
</tr>
<tr>
<td>JPEG_HUFFENC_DCx_y</td>
<td></td>
<td>1425</td>
</tr>
<tr>
<td>JPEG_HUFFMINx_y</td>
<td></td>
<td>1421-1422</td>
</tr>
<tr>
<td>JPEG_HUFFSYMx_y</td>
<td></td>
<td>1423</td>
</tr>
<tr>
<td>JPEG_QMEMx_y</td>
<td></td>
<td>1421</td>
</tr>
<tr>
<td>JPEG_SR</td>
<td></td>
<td>1418</td>
</tr>
</tbody>
</table>

**Index**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPTIM_ARR</td>
<td></td>
<td>2112</td>
</tr>
<tr>
<td>LPTIM_CCMR1</td>
<td></td>
<td>2114</td>
</tr>
<tr>
<td>LPTIM_CCR1</td>
<td></td>
<td>2111</td>
</tr>
<tr>
<td>LPTIM_CCR2</td>
<td></td>
<td>2117</td>
</tr>
<tr>
<td>LPTIM_CFGFR</td>
<td></td>
<td>2107</td>
</tr>
<tr>
<td>LPTIM_CFGFR2</td>
<td></td>
<td>2113</td>
</tr>
<tr>
<td>LPTIM_CNT</td>
<td></td>
<td>2112</td>
</tr>
<tr>
<td>LPTIM_CR</td>
<td></td>
<td>2110</td>
</tr>
<tr>
<td>LPTIM_RCR</td>
<td></td>
<td>2114</td>
</tr>
<tr>
<td>LPUART_BRR</td>
<td></td>
<td>2544</td>
</tr>
<tr>
<td>LPUART_CR1</td>
<td></td>
<td>2530, 2533</td>
</tr>
<tr>
<td>LPUART_CR2</td>
<td></td>
<td>2536</td>
</tr>
<tr>
<td>LPUART_CR3</td>
<td></td>
<td>2538, 2541</td>
</tr>
<tr>
<td>LPUART_ICR</td>
<td></td>
<td>2552</td>
</tr>
<tr>
<td>LPUART_ISR</td>
<td></td>
<td>2545, 2549</td>
</tr>
<tr>
<td>LPUART_PRES</td>
<td></td>
<td>2554</td>
</tr>
<tr>
<td>LPTIMx_DIER</td>
<td></td>
<td>2102, 2104-2105</td>
</tr>
<tr>
<td>LPTIMx_ICR</td>
<td></td>
<td>2099-2101</td>
</tr>
<tr>
<td>LPTIMx_ISR</td>
<td></td>
<td>2094-2095, 2097</td>
</tr>
<tr>
<td>LPTIM_CFGR2</td>
<td></td>
<td>2113</td>
</tr>
<tr>
<td>LPTIM_CFGR</td>
<td></td>
<td>2107</td>
</tr>
<tr>
<td>LPTIM_CFGR1</td>
<td></td>
<td>2111</td>
</tr>
<tr>
<td>JPEG_CONFR0</td>
<td></td>
<td>1413</td>
</tr>
<tr>
<td>JPEG_CONFR1</td>
<td></td>
<td>1414</td>
</tr>
<tr>
<td>JPEG_CONFR2</td>
<td></td>
<td>1415</td>
</tr>
<tr>
<td>JPEG_CONFR3</td>
<td></td>
<td>1415</td>
</tr>
</tbody>
</table>

*STM*
| LTDC_LxCKCR | 1398 |
| LTDC_LxCLUTWR | 1402 |
| LTDC_LxCR | 1395 |
| LTDC_LxDCCR | 1399 |
| LTDC_LxFCCR | 1398 |
| LTDC_LxWHPCR | 1396 |
| LTDC_LxWPCR | 1397 |
| LTDC_SRCR | 1391 |
| LTDC_SSCR | 1387 |
| LTDC_TWCR | 1389 |

M
<p>| M7_CPUROM_CIDR0 | 3703 |
| M7_CPUROM_CIDR1 | 3703 |
| M7_CPUROM_CIDR2 | 3704 |
| M7_CPUROM_CIDR3 | 3704 |
| M7_CPUROM_MEMTYPE | 3701 |
| M7_CPUROM_PIDR0 | 3701 |
| M7_CPUROM_PIDR1 | 3702 |
| M7_CPUROM_PIDR2 | 3702 |
| M7_CPUROM_PIDR3 | 3703 |
| M7_CPUROM_PIDR4 | 3701 |
| M7_DWT_CIDR0 | 3720 |
| M7_DWT_CIDR1 | 3720 |
| M7_DWT_CIDR2 | 3720 |
| M7_DWT_CIDR3 | 3721 |
| M7_DWT_COMPx | 3716 |
| M7_DWT_CPICNT | 3714 |
| M7_DWT_CTRL | 3712 |
| M7_DWT_CYCCNT | 3713 |
| M7_DWT_EXCCNT | 3714 |
| M7_DWT_FOLDCNT | 3715 |
| M7_DWT_FUNCTx | 3716 |
| M7_DWT_LSCNT | 3715 |
| M7_DWT_MASKx | 3716 |
| M7_DWT_PCSR | 3715 |
| M7_DWT_PIDR0 | 3718 |
| M7_DWT_PIDR1 | 3718 |
| M7_DWT_PIDR2 | 3719 |
| M7_DWT_PIDR3 | 3719 |
| M7_DWT_PIDR4 | 3718 |
| M7_DWT_SLPCNT | 3714 |
| M7_ETM_AUTHSTAT | 3760 |
| M7_ETM_CCCTL | 3744 |
| M7_ETM_CIDR0 | 3763 |
| M7_ETM_CIDR1 | 3763 |
| M7_ETM_CIDR2 | 3764 |
| M7_ETM_CIDR3 | 3764 |
| M7_ETM_CLAIMCLR | 3758 |
| M7_ETM_CLAIMSET | 3758 |
| M7_ETM_CNTRLDV | 3747 |
| M7_ETM_CONFIG | 3740 |
| M7_ETM_DEVARCH | 3760 |
| M7_ETM_DEVTYPE | 3761 |
| M7_ETM_EVENTCTL0 | 3741 |
| M7_ETM_EVENTCTL1 | 3742 |
| M7_ETM_IDR0 | 3750 |
| M7_ETM_IDR1 | 3751 |
| M7_ETM_IDR10 | 3748 |
| M7_ETM_IDR11 | 3748 |
| M7_ETM_IDR12 | 3749 |
| M7_ETM_IDR13 | 3749 |
| M7_ETM_IDR2 | 3751 |
| M7_ETM_IDR3 | 3752 |
| M7_ETM_IDR4 | 3753 |
| M7_ETM_IDR5 | 3753 |
| M7_ETM_IDR8 | 3747 |
| M7_ETM_IDR9 | 3748 |
| M7_ETM_IMSPEC0 | 3749 |
| M7_ETM_LAR | 3759 |
| M7_ETM_LSR | 3759 |
| M7_ETM_PDC | 3757 |
| M7_ETM_PED | 3757 |
| M7_ETM_PDS | 3757 |
| M7_ETM_PIDR0 | 3761 |
| M7_ETM_PIDR1 | 3762 |
| M7_ETM_PIDR2 | 3762 |
| M7_ETM_PIDR3 | 3763 |
| M7_ETM_PIDR4 | 3761 |
| M7_ETM_PRCGLT | 3739 |
| M7_ETM_PROCSEL | 3739 |
| M7_ETM_RSCCTL2 | 3754 |
| M7_ETM_RSCCTL3 | 3755 |
| M7_ETM_SSCC0 | 3755 |
| M7_ETM_SSSC0 | 3756 |
| M7_ETM_SSPCIC0 | 3756 |
| M7_ETM_STALLCTL | 3742 |
| M7_ETM_STAT | 3739 |
| M7_ETM_SYNCP | 3743 |
| M7_ETM_TRACEID | 3744 |
| M7_ETM_TSCTL | 3743 |
| M7_ETM_VICTL | 3745 |
| M7_ETM_VIPCSSCTL | 3746 |
| M7_ETM_VISSCTL | 3746 |
| M7_FPB_CIDR0 | 3735 |
| M7_FPB_CIDR1 | 3735 |
| M7_FPB_CIDR2 | 3736 |
| M7_FPB_CIDR3 | 3736 |
| M7_FPB_COMPx | 3733 |
| M7_FPB_CTRL | 3732 |
| M7_FPB_PIDR0 | 3733 |
| M7_FPB_PIDR1 | 3734 |
| M7_FPB_PIDR2 | 3734 |
| M7_FPB_PIDR3 | 3735 |</p>
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0477</th>
</tr>
</thead>
<tbody>
<tr>
<td>OTG_HCTSIZx</td>
<td>2965, 3126</td>
</tr>
<tr>
<td>OTG_HFIR</td>
<td>2956, 3116</td>
</tr>
<tr>
<td>OTG_HNUM</td>
<td>2957, 3117</td>
</tr>
<tr>
<td>OTG_HNPTXFSIZ</td>
<td>2947, 3107</td>
</tr>
<tr>
<td>OTG_HNPTXSTS</td>
<td>2948, 3108</td>
</tr>
<tr>
<td>OTG_HPRT</td>
<td>2960, 3120</td>
</tr>
<tr>
<td>OTG_HPTXFSSIZ</td>
<td>2955, 3115</td>
</tr>
<tr>
<td>OTG_HPTXSTS</td>
<td>2958, 3118</td>
</tr>
<tr>
<td>OTG_HS_DIEPEACHMSK1</td>
<td>2976</td>
</tr>
<tr>
<td>OTG_HS_DOEPEACHMSK1</td>
<td>2977</td>
</tr>
<tr>
<td>OTG_PCGCCTL</td>
<td>2996, 3126</td>
</tr>
<tr>
<td>PSSI_ICR</td>
<td>1373</td>
</tr>
<tr>
<td>PSSI_DR</td>
<td>1373</td>
</tr>
<tr>
<td>PSSI_MIS</td>
<td>1372</td>
</tr>
<tr>
<td>PSSI_RIS</td>
<td>1371</td>
</tr>
<tr>
<td>PSSI_SR</td>
<td>1370</td>
</tr>
<tr>
<td>PWR_APCR</td>
<td>331</td>
</tr>
<tr>
<td>PWR_CR1</td>
<td>317</td>
</tr>
<tr>
<td>PWR_CSR1</td>
<td>320</td>
</tr>
<tr>
<td>PWR_CSR2</td>
<td>322</td>
</tr>
<tr>
<td>PWR_CSR3</td>
<td>325</td>
</tr>
<tr>
<td>PWR_CSR4</td>
<td>326</td>
</tr>
<tr>
<td>PWR_PDCRN</td>
<td>334</td>
</tr>
<tr>
<td>PWR_PDCRO</td>
<td>336</td>
</tr>
<tr>
<td>PWR_PDCRP</td>
<td>337</td>
</tr>
<tr>
<td>PWR_PUCRN</td>
<td>333</td>
</tr>
<tr>
<td>PWR_PUCRO</td>
<td>335</td>
</tr>
<tr>
<td>PWR_SR1</td>
<td>319</td>
</tr>
<tr>
<td>PWR_UCPDR</td>
<td>330</td>
</tr>
<tr>
<td>PWR_WKUPCR</td>
<td>327</td>
</tr>
<tr>
<td>PWR_WKUPEPR</td>
<td>329</td>
</tr>
<tr>
<td>PWR_WKUPFR</td>
<td>328</td>
</tr>
<tr>
<td>P</td>
<td>1627</td>
</tr>
<tr>
<td>PKA_CLRFR</td>
<td>1624</td>
</tr>
<tr>
<td>PKA_CR</td>
<td>1626</td>
</tr>
<tr>
<td>PSSI_CR</td>
<td>1368</td>
</tr>
<tr>
<td>PSSI_DR</td>
<td>1373</td>
</tr>
<tr>
<td>PSSI_ICR</td>
<td>1373</td>
</tr>
<tr>
<td>PSSI_IER</td>
<td>1372</td>
</tr>
<tr>
<td>PSSI_MIS</td>
<td>1372</td>
</tr>
<tr>
<td>PSSI_RIS</td>
<td>1371</td>
</tr>
<tr>
<td>PSSI_SR</td>
<td>1370</td>
</tr>
<tr>
<td>R</td>
<td>170, 176</td>
</tr>
<tr>
<td>RAMECC_IER</td>
<td>171</td>
</tr>
<tr>
<td>RAMECC_MxCR</td>
<td>173, 176</td>
</tr>
<tr>
<td>RAMECC_MxFAR</td>
<td>174, 176</td>
</tr>
<tr>
<td>RAMECC_MxFDRH</td>
<td>174, 176</td>
</tr>
<tr>
<td>RAMECC_MxFDRL</td>
<td>173, 176</td>
</tr>
<tr>
<td>RAMECC_MxFECR</td>
<td>173, 176</td>
</tr>
<tr>
<td>RAMECC_MxSR</td>
<td>173, 176</td>
</tr>
<tr>
<td>RCC_AHB1ENR</td>
<td>496</td>
</tr>
<tr>
<td>RCC_AHB1LPENR</td>
<td>515</td>
</tr>
<tr>
<td>RCC_AHB1RSTR</td>
<td>474</td>
</tr>
<tr>
<td>RCC_AHB2ENR</td>
<td>498</td>
</tr>
<tr>
<td>RCC_AHB2LPENR</td>
<td>517</td>
</tr>
<tr>
<td>RCC_AHB2RSTR</td>
<td>475</td>
</tr>
<tr>
<td>RCC_AHB3ENR</td>
<td>499</td>
</tr>
<tr>
<td>RCC_AHB3LPENR</td>
<td>500</td>
</tr>
<tr>
<td>RCC_AHB4ENR</td>
<td>477</td>
</tr>
<tr>
<td>RCC_AHB4RSTR</td>
<td>519</td>
</tr>
<tr>
<td>RCC_AHB5ENR</td>
<td>502</td>
</tr>
<tr>
<td>RCC_AHB5LPENR</td>
<td>513</td>
</tr>
<tr>
<td>RCC_AHB5RSTR</td>
<td>479</td>
</tr>
<tr>
<td>RCC_APB1ENR1</td>
<td>504</td>
</tr>
<tr>
<td>RCC_APB1ENR2</td>
<td>507</td>
</tr>
<tr>
<td>RCC_APB1LPENR1</td>
<td>521</td>
</tr>
<tr>
<td>RCC_APB1LPENR2</td>
<td>524</td>
</tr>
<tr>
<td>RCC_APB1RSTR1</td>
<td>481</td>
</tr>
<tr>
<td>RCC_APB1RSTR2</td>
<td>484</td>
</tr>
<tr>
<td>RCC_APB2ENR</td>
<td>508</td>
</tr>
<tr>
<td>RCC_APB2LPENR</td>
<td>525</td>
</tr>
<tr>
<td>RCC_APB2RSTR</td>
<td>485</td>
</tr>
<tr>
<td>RCC_APB4ENR</td>
<td>509</td>
</tr>
<tr>
<td>RCC_APB4LPENR</td>
<td>527</td>
</tr>
<tr>
<td>RCC_APB4RSTR</td>
<td>487</td>
</tr>
<tr>
<td>RCC_APB5ENR</td>
<td>512</td>
</tr>
<tr>
<td>RCC_APB5LPENR</td>
<td>529</td>
</tr>
<tr>
<td>RCC_APB5RSTR</td>
<td>489</td>
</tr>
<tr>
<td>RCC_APBCFGR</td>
<td>435</td>
</tr>
<tr>
<td>RCC_BDCR</td>
<td>471</td>
</tr>
<tr>
<td>RCC_BMCFGR</td>
<td>434</td>
</tr>
<tr>
<td>RCC_CCIPR1</td>
<td>457</td>
</tr>
<tr>
<td>RCC_CCIPR2</td>
<td>460</td>
</tr>
<tr>
<td>RCC_CCIPR3</td>
<td>462</td>
</tr>
<tr>
<td>RCC_CCIPR4</td>
<td>464</td>
</tr>
<tr>
<td>RCC_CDCFGR</td>
<td>433</td>
</tr>
<tr>
<td>RCC_CFG</td>
<td>430</td>
</tr>
<tr>
<td>RCC_CICR</td>
<td>469</td>
</tr>
<tr>
<td>RCC_CIER</td>
<td>466</td>
</tr>
<tr>
<td>RCC_CIFR</td>
<td>467</td>
</tr>
<tr>
<td>RCC_CKGDISR</td>
<td>490</td>
</tr>
<tr>
<td>RCC_CKPROTR</td>
<td>427</td>
</tr>
<tr>
<td>RCC_CR</td>
<td>423</td>
</tr>
<tr>
<td>RCC_CRRCR</td>
<td>429</td>
</tr>
<tr>
<td>RCC_CSR</td>
<td>429</td>
</tr>
<tr>
<td>RCC_HSICFG</td>
<td>473</td>
</tr>
<tr>
<td>RCC_PLL1DIVR1</td>
<td>442</td>
</tr>
<tr>
<td>RCC_PLL1DIVR2</td>
<td>444</td>
</tr>
<tr>
<td>RCC_PLL1FRACR</td>
<td>444</td>
</tr>
<tr>
<td>RCC_PLL1SSCGR</td>
<td>446</td>
</tr>
<tr>
<td>RCC_PLL2DIVR1</td>
<td>447</td>
</tr>
<tr>
<td>RCC_PLL2DIVR2</td>
<td>SAES_KEYR1</td>
</tr>
<tr>
<td>RCC_PLL2FRACR</td>
<td>SAES_KEYR2</td>
</tr>
<tr>
<td>RCC_PLL2SSCGR</td>
<td>SAES_KEYR3</td>
</tr>
<tr>
<td>RCC_PLL3DIVR1</td>
<td>SAES_KEYR4</td>
</tr>
<tr>
<td>RCC_PLL3DIVR2</td>
<td>SAES_KEYR5</td>
</tr>
<tr>
<td>RCC_PLL3FRACR</td>
<td>SAES_KEYR6</td>
</tr>
<tr>
<td>RCC_PLL3SSCGR</td>
<td>SAES_KEYR7</td>
</tr>
<tr>
<td>RCC_PLLCFGFR</td>
<td>SAES_SR</td>
</tr>
<tr>
<td>RCC_PLLCKSELR</td>
<td>SAES_SUSPxR</td>
</tr>
<tr>
<td>RCC_RSR</td>
<td>SAI_ACLRFR</td>
</tr>
<tr>
<td>RNG_CR</td>
<td>SAI_ACR1</td>
</tr>
<tr>
<td>RNG_DR</td>
<td>SAI_ACR2</td>
</tr>
<tr>
<td>RNG_HTCR</td>
<td>SAI_ADR</td>
</tr>
<tr>
<td>RNG_NSCR</td>
<td>SAI_AFRCR</td>
</tr>
<tr>
<td>RNG_SR</td>
<td>SAI_AIM</td>
</tr>
<tr>
<td>RTC_ALRABINR</td>
<td>SAI_ASLOTR</td>
</tr>
<tr>
<td>RTC_ALRBINR</td>
<td>SAI_ASR</td>
</tr>
<tr>
<td>RTC_ALRMAR</td>
<td>SAI_BCLRFR</td>
</tr>
<tr>
<td>RTC_ALRMASSR</td>
<td>SAI_BCR1</td>
</tr>
<tr>
<td>RTC_ALRMBR</td>
<td>SAI_BCR2</td>
</tr>
<tr>
<td>RTC_ALRMBSSR</td>
<td>SAI_BDR</td>
</tr>
<tr>
<td>RTC_CALR</td>
<td>SAI_BFRCR</td>
</tr>
<tr>
<td>RTC_CR</td>
<td>SAI_BIM</td>
</tr>
<tr>
<td>RTC_DR</td>
<td>SAI_BSLOTR</td>
</tr>
<tr>
<td>RTC_ICSR</td>
<td>SAI_BSR</td>
</tr>
<tr>
<td>RTC_MISR</td>
<td>SAI_GCR</td>
</tr>
<tr>
<td>RTC_PRER</td>
<td>SAI_PDMCR</td>
</tr>
<tr>
<td>RTC_PRIVCFGFR</td>
<td>SAI_PDMDLY</td>
</tr>
<tr>
<td>RTC_SCR</td>
<td>SBS_BKLOCKR</td>
</tr>
<tr>
<td>RTC_SHIFTR</td>
<td>SBS_BOOTSR</td>
</tr>
<tr>
<td>RTC_SR</td>
<td>SBS_CCCSR</td>
</tr>
<tr>
<td>RTC_SSR</td>
<td>SBS_CCSWVALR</td>
</tr>
<tr>
<td>RTC_TR</td>
<td>SBS_CCVALR</td>
</tr>
<tr>
<td>RTC_TSDR</td>
<td>SBSDBGCR</td>
</tr>
<tr>
<td>RTC_TSSSR</td>
<td>SBSDBGLOCKR</td>
</tr>
<tr>
<td>RTC_TSTR</td>
<td>SBS_EXTICRx</td>
</tr>
<tr>
<td>RTC_WPR</td>
<td>SBS_FPUIMR</td>
</tr>
<tr>
<td>RTC_WUTR</td>
<td>SBS_HDPLCR</td>
</tr>
</tbody>
</table>

**S**

<p>| SAES_CR | 1485 |
| SAES_DINR | 1489 |
| SAES_DOUTR | 1490 |
| SAES_ICR | 1497 |
| SAES_IER | 1495 |
| SAES_ISR | 1496 |
| SAES_IVR0 | 1492 |
| SAES_IVR1 | 1492 |
| SAES_IVR2 | 1492 |
| SAES_IVR3 | 1493 |
| SAES_KEYR0 | 1490 |</p>
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0477</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDMMC_IDMABAR</td>
<td>2825</td>
</tr>
<tr>
<td>SDMMC_IDMABASER</td>
<td>2824</td>
</tr>
<tr>
<td>SDMMC_IDMABSZIER</td>
<td>2823</td>
</tr>
<tr>
<td>SDMMC_IDMACCTLR</td>
<td>2822</td>
</tr>
<tr>
<td>SDMMC_IDMALAR</td>
<td>2824</td>
</tr>
<tr>
<td>SDMMC_MASKR</td>
<td>2819</td>
</tr>
<tr>
<td>SDMMC_POWER</td>
<td>2804</td>
</tr>
<tr>
<td>SDMMC_RESPCMDR</td>
<td>2809</td>
</tr>
<tr>
<td>SDMMC_RESPxR</td>
<td>2810</td>
</tr>
<tr>
<td>SDMMC_STAR</td>
<td>2814</td>
</tr>
<tr>
<td>SPDIFRX_CR</td>
<td>2724</td>
</tr>
<tr>
<td>SPDIFRX_CSR</td>
<td>2732</td>
</tr>
<tr>
<td>SPDIFRX_DIR</td>
<td>2732</td>
</tr>
<tr>
<td>SPDIFRX_FMT0_DR</td>
<td>2730</td>
</tr>
<tr>
<td>SPDIFRX_FMT1_DR</td>
<td>2730</td>
</tr>
<tr>
<td>SPDIFRX_FMT2_DR</td>
<td>2731</td>
</tr>
<tr>
<td>SPDIFRX_IFCR</td>
<td>2729</td>
</tr>
<tr>
<td>SPDIFRX_IMR</td>
<td>2726</td>
</tr>
<tr>
<td>SPDIFRX_SR</td>
<td>2727</td>
</tr>
<tr>
<td>SPI_CFG1</td>
<td>2616</td>
</tr>
<tr>
<td>SPI_CFG2</td>
<td>2619</td>
</tr>
<tr>
<td>SPI_CR1</td>
<td>2613</td>
</tr>
<tr>
<td>SPI_CR2</td>
<td>2615</td>
</tr>
<tr>
<td>SPI_CRCPOLY</td>
<td>2627</td>
</tr>
<tr>
<td>SPI_I2SCFGR</td>
<td>2630</td>
</tr>
<tr>
<td>SPI_IER</td>
<td>2622</td>
</tr>
<tr>
<td>SPI_IFCR</td>
<td>2625</td>
</tr>
<tr>
<td>SPI_RXCRC</td>
<td>2629</td>
</tr>
<tr>
<td>SPI_RXDR</td>
<td>2627</td>
</tr>
<tr>
<td>SPI_SR</td>
<td>2623</td>
</tr>
<tr>
<td>SPI_TXCRC</td>
<td>2628</td>
</tr>
<tr>
<td>SPI_TXDR</td>
<td>2626</td>
</tr>
<tr>
<td>SPI_UDRDR</td>
<td>2629</td>
</tr>
<tr>
<td>SWO_AUTHORSTAT</td>
<td>3672</td>
</tr>
<tr>
<td>SWO_CIDR0</td>
<td>3676</td>
</tr>
<tr>
<td>SWO_CIDR1</td>
<td>3676</td>
</tr>
<tr>
<td>SWO_CIDR2</td>
<td>3677</td>
</tr>
<tr>
<td>SWO_CIDR3</td>
<td>3677</td>
</tr>
<tr>
<td>SWO_CLAIMCLCR</td>
<td>3671</td>
</tr>
<tr>
<td>SWO_CLAIMSET</td>
<td>3670</td>
</tr>
<tr>
<td>SWO_CODR</td>
<td>3669</td>
</tr>
<tr>
<td>SWO_DEVID</td>
<td>3673</td>
</tr>
<tr>
<td>SWO_DEVTYPE</td>
<td>3673</td>
</tr>
<tr>
<td>SWO_FFSR</td>
<td>3670</td>
</tr>
<tr>
<td>SWO_LAR</td>
<td>3671</td>
</tr>
<tr>
<td>SWO_LSR</td>
<td>3672</td>
</tr>
<tr>
<td>SWO_PIDR0</td>
<td>3674</td>
</tr>
<tr>
<td>SWO_PIDR1</td>
<td>3675</td>
</tr>
<tr>
<td>SWO_PIDR2</td>
<td>3675</td>
</tr>
<tr>
<td>SWO_PIDR3</td>
<td>3676</td>
</tr>
<tr>
<td>SWO_PIDR4</td>
<td>3674</td>
</tr>
<tr>
<td>SWO_SPPR</td>
<td>3669</td>
</tr>
<tr>
<td>SYSROM_CIDR0</td>
<td>3593</td>
</tr>
<tr>
<td>SYSROM_CIDR1</td>
<td>3593</td>
</tr>
<tr>
<td>SYSROM_CIDR2</td>
<td>3594</td>
</tr>
<tr>
<td>SYSROM_CIDR3</td>
<td>3594</td>
</tr>
<tr>
<td>SYSROM_MEMTYPE</td>
<td>3591</td>
</tr>
<tr>
<td>SYSROM_PIDR0</td>
<td>3591</td>
</tr>
<tr>
<td>SYSROM_PIDR1</td>
<td>3592</td>
</tr>
<tr>
<td>SYSROM_PIDR2</td>
<td>3592</td>
</tr>
<tr>
<td>SYSROM_PIDR3</td>
<td>3593</td>
</tr>
<tr>
<td>SYSROM_PIDR4</td>
<td>3591</td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
<tr>
<td>TAMP_ATCR1</td>
<td>2213</td>
</tr>
<tr>
<td>TAMP_ATCR2</td>
<td>2217</td>
</tr>
<tr>
<td>TAMP_ATOR</td>
<td>2216</td>
</tr>
<tr>
<td>TAMP_ATSEEDR</td>
<td>2216</td>
</tr>
<tr>
<td>TAMP_BKPxR</td>
<td>2231</td>
</tr>
<tr>
<td>TAMP_CFGR</td>
<td>2220</td>
</tr>
<tr>
<td>TAMP_COUNT1R</td>
<td>2230</td>
</tr>
<tr>
<td>TAMP_CR1</td>
<td>2206</td>
</tr>
<tr>
<td>TAMP_CR2</td>
<td>2208</td>
</tr>
<tr>
<td>TAMP_CR3</td>
<td>2210</td>
</tr>
<tr>
<td>TAMP_FLTCR</td>
<td>2212</td>
</tr>
<tr>
<td>TAMP_IER</td>
<td>2222</td>
</tr>
<tr>
<td>TAMP_MISR</td>
<td>2226</td>
</tr>
<tr>
<td>TAMP_PRIVCFGR</td>
<td>2221</td>
</tr>
<tr>
<td>TAMP_RPCFGR</td>
<td>2230</td>
</tr>
<tr>
<td>TAMP_SCR</td>
<td>2228</td>
</tr>
<tr>
<td>TAMP_SR</td>
<td>2224</td>
</tr>
<tr>
<td>TIM1_AF1</td>
<td>1760</td>
</tr>
<tr>
<td>TIM1_AF2</td>
<td>1763</td>
</tr>
<tr>
<td>TIM1_ARR</td>
<td>1746</td>
</tr>
<tr>
<td>TIM1_BDTR</td>
<td>1750</td>
</tr>
<tr>
<td>TIM1_CCER</td>
<td>1741</td>
</tr>
<tr>
<td>TIM1_CCMR1</td>
<td>1732, 1734</td>
</tr>
<tr>
<td>TIM1_CCMR2</td>
<td>1737-1738</td>
</tr>
<tr>
<td>TIM1_CCMR3</td>
<td>1756</td>
</tr>
<tr>
<td>TIM1_CCR1</td>
<td>1747</td>
</tr>
<tr>
<td>TIM1_CCR2</td>
<td>1747</td>
</tr>
<tr>
<td>TIM1_CCR3</td>
<td>1748</td>
</tr>
<tr>
<td>TIM1_CCR4</td>
<td>1749</td>
</tr>
<tr>
<td>TIM1_CCR5</td>
<td>1754</td>
</tr>
<tr>
<td>TIM1_CCR6</td>
<td>1755</td>
</tr>
<tr>
<td>TIM1_CNT</td>
<td>1745</td>
</tr>
<tr>
<td>TIM1_CR1</td>
<td>1718</td>
</tr>
<tr>
<td>TIM1_CR2</td>
<td>1719</td>
</tr>
<tr>
<td>TIM1_DCR</td>
<td>1765</td>
</tr>
<tr>
<td>TIM1_DIER</td>
<td>1727</td>
</tr>
<tr>
<td>TIM1_DMAR</td>
<td>1767</td>
</tr>
<tr>
<td>TIM1_DTR2</td>
<td>1757</td>
</tr>
<tr>
<td>TIM1_ECR</td>
<td>1758</td>
</tr>
<tr>
<td>RM0477</td>
<td>Index</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>TIM1_EGR</td>
<td>1731</td>
</tr>
<tr>
<td>TIM1_PSC</td>
<td>1745</td>
</tr>
<tr>
<td>TIM1_RCR</td>
<td>1746</td>
</tr>
<tr>
<td>TIM1_SMCR</td>
<td>1723</td>
</tr>
<tr>
<td>TIM1_SR</td>
<td>1728</td>
</tr>
<tr>
<td>TIM1_TISEL</td>
<td>1759</td>
</tr>
<tr>
<td>TIM15_AF1</td>
<td>2037</td>
</tr>
<tr>
<td>TIM15_AF2</td>
<td>2039</td>
</tr>
<tr>
<td>TIM15_ARR</td>
<td>2030</td>
</tr>
<tr>
<td>TIM15_BDTR</td>
<td>2032</td>
</tr>
<tr>
<td>TIM15_CCR1</td>
<td>2031</td>
</tr>
<tr>
<td>TIM15_CCR2</td>
<td>2032</td>
</tr>
<tr>
<td>TIM15_CNT</td>
<td>2029</td>
</tr>
<tr>
<td>TIM15_CR1</td>
<td>2013</td>
</tr>
<tr>
<td>TIM15_CR2</td>
<td>2014</td>
</tr>
<tr>
<td>TIM15_DCR</td>
<td>2040</td>
</tr>
<tr>
<td>TIM15_DIER</td>
<td>2018</td>
</tr>
<tr>
<td>TIM15_DMAR</td>
<td>2041</td>
</tr>
<tr>
<td>TIM15_DTR2</td>
<td>2035</td>
</tr>
<tr>
<td>TIM15_EGR</td>
<td>2021</td>
</tr>
<tr>
<td>TIM15_PSC</td>
<td>2029</td>
</tr>
<tr>
<td>TIM15_RCR</td>
<td>2030</td>
</tr>
<tr>
<td>TIM15_SMR</td>
<td>2016</td>
</tr>
<tr>
<td>TIM15_SR</td>
<td>2019</td>
</tr>
<tr>
<td>TIM15_TISEL</td>
<td>2036</td>
</tr>
<tr>
<td>TIMx_AF1</td>
<td>1876, 2062</td>
</tr>
<tr>
<td>TIMx_AF2</td>
<td>1877, 2065</td>
</tr>
<tr>
<td>TIMx_ARR</td>
<td>1870, 1900, 1948, 1961, 2056</td>
</tr>
<tr>
<td>TIMx_BDTR</td>
<td>2058</td>
</tr>
<tr>
<td>TIMx_CCER</td>
<td>1867, 1946, 1959, 2032</td>
</tr>
<tr>
<td>TIMx_CCMR1</td>
<td>1859, 1861, 1942-1943, 1956-1957, 2049-2050</td>
</tr>
<tr>
<td>TIMx_CCMR2</td>
<td>1863-1864</td>
</tr>
<tr>
<td>TIMx_CCR1</td>
<td>1870, 1949, 1962, 2057</td>
</tr>
<tr>
<td>TIMx_CCR2</td>
<td>1871, 1949</td>
</tr>
<tr>
<td>TIMx_CCR3</td>
<td>1872</td>
</tr>
<tr>
<td>TIMx_CCR4</td>
<td>1873</td>
</tr>
<tr>
<td>TIMx_CNT</td>
<td>1869, 1899, 1947, 1960, 2055</td>
</tr>
<tr>
<td>TIMx_CR1</td>
<td>1848, 1896, 1935, 1953, 2044</td>
</tr>
<tr>
<td>TIMx_CR2</td>
<td>1849, 1898, 1936, 2045</td>
</tr>
<tr>
<td>TIMx_DCR</td>
<td>1878, 2065</td>
</tr>
<tr>
<td>TIMx_DIER</td>
<td>1855, 1898, 1939, 1954, 2046</td>
</tr>
<tr>
<td>TIMx_DMAR</td>
<td>1879, 2066</td>
</tr>
<tr>
<td>TIMx_DTR2</td>
<td>2061</td>
</tr>
<tr>
<td>TIMx_ECR</td>
<td>1874</td>
</tr>
<tr>
<td>TIMx_EGR</td>
<td>1858, 1899, 1941, 1955, 2048</td>
</tr>
<tr>
<td>TIMx_PSC</td>
<td>1869, 1900, 1948, 1961, 2055</td>
</tr>
<tr>
<td>TIMx_RCR</td>
<td>2056</td>
</tr>
<tr>
<td>TIMx_SMCR</td>
<td>1851, 1937</td>
</tr>
<tr>
<td>TIMx_SR</td>
<td>1856, 1899, 1940, 1954, 2047</td>
</tr>
<tr>
<td>TIMx_TISEL</td>
<td>1875, 1950, 1962, 2062</td>
</tr>
<tr>
<td>TPIU_AUTHSTAT</td>
<td>3660</td>
</tr>
<tr>
<td>TPIU_CIDR0</td>
<td>3663</td>
</tr>
<tr>
<td>TPIU_CIDR1</td>
<td>3664</td>
</tr>
<tr>
<td>TPIU_CIDR2</td>
<td>3664</td>
</tr>
<tr>
<td>TPIU_CIDR3</td>
<td>3664</td>
</tr>
<tr>
<td>TPIU_CLAIMCLR</td>
<td>3658</td>
</tr>
<tr>
<td>TPIU_CLAIMSET</td>
<td>3658</td>
</tr>
<tr>
<td>TPIU_CURPSIZE</td>
<td>3651</td>
</tr>
<tr>
<td>TPIU_CURTPM</td>
<td>3654</td>
</tr>
<tr>
<td>TPIU_DEVID</td>
<td>3660</td>
</tr>
<tr>
<td>TPIU_DEVTYPE</td>
<td>3661</td>
</tr>
<tr>
<td>TPIU_FFCCR</td>
<td>3656</td>
</tr>
<tr>
<td>TPIU_FFSR</td>
<td>3655</td>
</tr>
<tr>
<td>TPIU_FSCCR</td>
<td>3657</td>
</tr>
<tr>
<td>TPIU_LAR</td>
<td>3659</td>
</tr>
<tr>
<td>TPIU_LSFR</td>
<td>3659</td>
</tr>
<tr>
<td>TPIU_PIDR0</td>
<td>3662</td>
</tr>
<tr>
<td>TPIU_PIDR1</td>
<td>3662</td>
</tr>
<tr>
<td>TPIU_PIDR2</td>
<td>3662</td>
</tr>
<tr>
<td>TPIU_PIDR3</td>
<td>3663</td>
</tr>
<tr>
<td>TPIU_PIDR4</td>
<td>3661</td>
</tr>
<tr>
<td>TPIU_SUPPSIZE</td>
<td>3651</td>
</tr>
<tr>
<td>TPIU_SUPTPM</td>
<td>3653</td>
</tr>
<tr>
<td>TPIU_SUPTRGM</td>
<td>3651</td>
</tr>
<tr>
<td>TPIU_TRGCNT</td>
<td>3652</td>
</tr>
<tr>
<td>TPIU_TRGMULT</td>
<td>3653</td>
</tr>
</tbody>
</table>

**U**

<p>| UCPD_CFGDR1 | 3243 |
| UCPD_CFGDR2 | 3245 |
| UCPD_CFGDR3 | 3246 |
| UCPD_CR | 3246 |
| UCPD_ICR | 3253 |
| UCPD_IMR | 3249 |
| UCPD_RX_ORDEXT1 | 3258 |
| UCPD_RX_ORDEXT2 | 3258 |
| UCPD_RX_ORDSET1 | 3256 |
| UCPD_RX_PAYSZR | 3257 |
| UCPD_RXDR | 3257 |
| UCPD_SR | 3250 |
| UCPD_TX_ORDSET1 | 3254 |
| UCPD_TX_PAYSZR | 3255 |
| UCPD_TXDR | 3255 |
| USART_BRR | 2478 |
| USART_CR1 | 2459, 2463 |
| USART_CR2 | 2466 |
| USART_CR3 | 2470, 2475 |
| USART_GTPR | 2479 |
| USART_ICR | 2493 |</p>
<table>
<thead>
<tr>
<th>Index</th>
<th>RM0477</th>
</tr>
</thead>
<tbody>
<tr>
<td>USART_ISR</td>
<td>2482, 2488</td>
</tr>
<tr>
<td>USART_PRESC</td>
<td>2496</td>
</tr>
<tr>
<td>USART_RDR</td>
<td>2495</td>
</tr>
<tr>
<td>USART_RQR</td>
<td>2481</td>
</tr>
<tr>
<td>USART_RTOR</td>
<td>2480</td>
</tr>
<tr>
<td>USART_TDR</td>
<td>2495</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td>VREFBUF_CCR</td>
</tr>
<tr>
<td>VREFBUF_CSR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>WWDG_CFR</td>
</tr>
<tr>
<td>WWDG_CR</td>
</tr>
<tr>
<td>WWDG_SR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>XYZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSPI_ABR</td>
</tr>
<tr>
<td>XSPI_AR</td>
</tr>
<tr>
<td>XSPI_CALFCR</td>
</tr>
<tr>
<td>XSPI_CALMR</td>
</tr>
<tr>
<td>XSPI_CALSIR</td>
</tr>
<tr>
<td>XSPI_CALSOR</td>
</tr>
<tr>
<td>XSPI_CCR</td>
</tr>
<tr>
<td>XSPI_CR</td>
</tr>
<tr>
<td>XSPI_DCR1</td>
</tr>
<tr>
<td>XSPI_DCR2</td>
</tr>
<tr>
<td>XSPI_DCR3</td>
</tr>
<tr>
<td>XSPI_DCR4</td>
</tr>
<tr>
<td>XSPI_DLR</td>
</tr>
<tr>
<td>XSPI_DR</td>
</tr>
<tr>
<td>XSPI_FCR</td>
</tr>
<tr>
<td>XSPI_HLCR</td>
</tr>
<tr>
<td>XSPI_IR</td>
</tr>
<tr>
<td>XSPI_LPTR</td>
</tr>
<tr>
<td>XSPI_PIR</td>
</tr>
<tr>
<td>XSPI_PSMAR</td>
</tr>
<tr>
<td>XSPI_PSMKR</td>
</tr>
<tr>
<td>XSPI_SR</td>
</tr>
<tr>
<td>XSPI_TCR</td>
</tr>
<tr>
<td>XSPI_WABR</td>
</tr>
<tr>
<td>XSPI_WCCR</td>
</tr>
<tr>
<td>XSPI_WIR</td>
</tr>
<tr>
<td>XSPI_WPABR</td>
</tr>
<tr>
<td>XSPI_WPCCR</td>
</tr>
<tr>
<td>XSPI_WPPIR</td>
</tr>
<tr>
<td>XSPI_WPTCR</td>
</tr>
<tr>
<td>XSPI_WTCR</td>
</tr>
<tr>
<td>XSPIIM_CR</td>
</tr>
</tbody>
</table>
IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2024 STMicroelectronics – All rights reserved